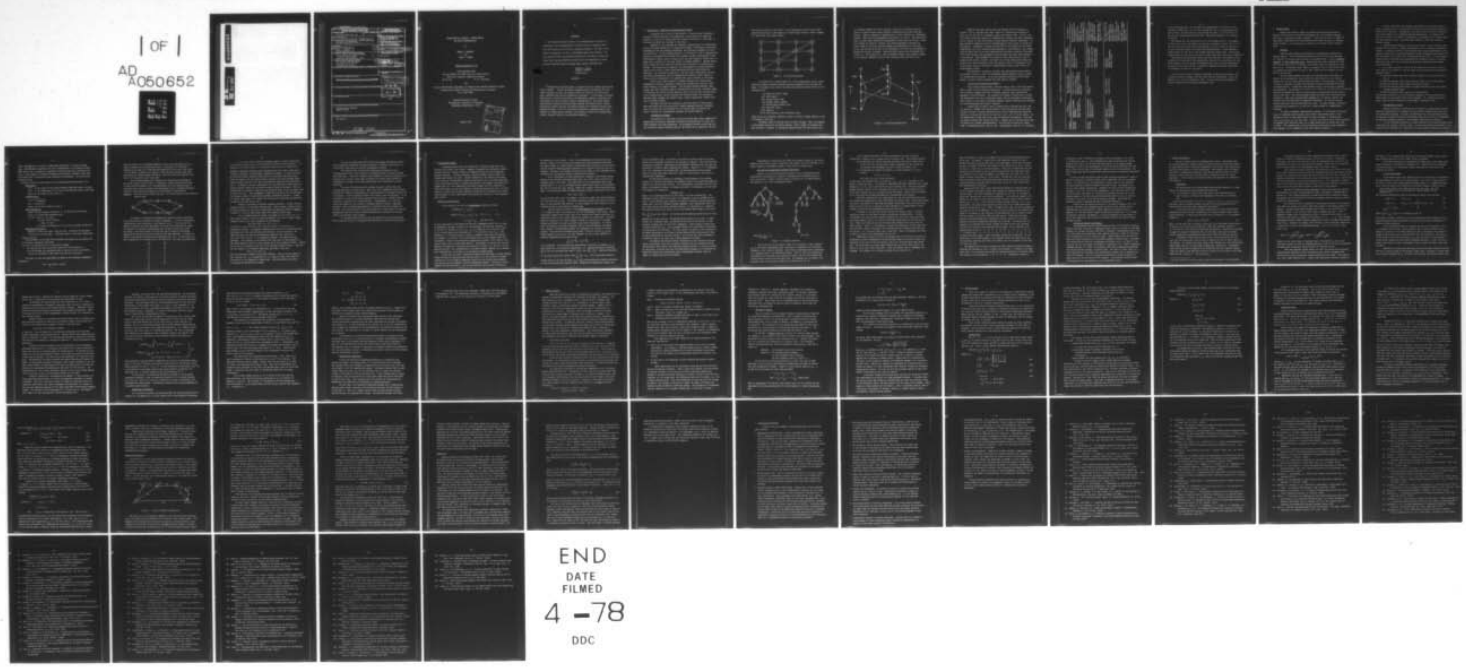
AD-A050 652 MASSACHUSETTS INST OF TECH CAMBRIDGE OPERATIONS RESE--ETC F/G 12/2
TRANSPORTATION PLANNING: NETWORK MODELS AND THEIR IMPLEMENTATIO--ETC(U)
JAN 78 T L MAGNANTI, B L GOLDEN N00014-75-C-0556
UNCLASSIFIED TR-143 NL



END
DATE FILMED
4 -78
DDC

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: Technical Report No. 143
2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER: (9)
4. TITLE (and Subtitle): (6) TRANSPORTATION PLANNING: NETWORK MODELS AND THEIR IMPLEMENTATION
5. TYPE OF REPORT & PERIOD COVERED: Technical Report, January 1978
6. PERFORMING ORG. REPORT NUMBER: (14) TR-143
7. AUTHOR(s): (10) Thomas L. Magnanti, Bruce L. Golden
8. CONTRACT OR GRANT NUMBER(s): (15) N0001 4-75-C-0556, DOT-TSC-1058
9. PERFORMING ORGANIZATION NAME AND ADDRESS: M.I.T. Operations Research Center, 77 Massachusetts Avenue, Cambridge, Massachusetts 02139
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: NR 347-027
11. CONTROLLING OFFICE NAME AND ADDRESS: O.R. Branch, ONR Navy Dept., 800 North Quincy Street, Arlington, VA 22217
12. REPORT DATE: Jan[redacted]78 (12)
13. NUMBER OF PAGES: 55 pages (59p.)
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)
15. SECURITY CLASS. (of this report)
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report): ~~Releasable without limitation on dissemination.~~

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

DDC
RECEIVED
MAR 1 1978
F

18. SUPPLEMENTARY NOTES
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Transportation Planning
Network Models

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

See page ii.

270 720

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TRANSPORTATION PLANNING: NETWORK MODELS AND THEIR IMPLEMENTATION

by

THOMAS L. MAGNANTI
and
BRUCE L. GOLDEN

Technical Report No. 143

Work Performed Under
Contract N00014-75-C-0556, Office of Naval Research
Multilevel Logistics Organization Models
NR 347-027 M.I.T. OSP 82491

and

Contract DOT-TSC-1058, Department of Transportation Advanced Research Program
Transportation Network Analysis and Decomposition Techniques
M.I.T. OSP 83185

Operations Research Center
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

January 1978

FOREWORD

The Operations Research Center at the Massachusetts Institute of Technology is an interdepartmental activity devoted to graduate education and research in the field of operations research. The work of the Center is supported, in part, by government grants and contracts. The work reported herein was supported (in part) by the Office of Naval Research under Contract N00014-75-C-0556 and by the Department of Transportation Advanced Research Program under contract DOT-TSC-1058.

Richard C. Larson
Jeremy F. Shapiro
Co-Directors

ABSTRACT

Transportation planning plays an essential role in shaping regional and urban lifestyle. Complex decisions regarding policy alternatives for railroads, shipping, airline, and roadway traffic can often be, and often have been, analyzed using network optimization techniques. In this paper, we survey applications of network algorithms to transportation planning, stressing network models and their efficient computer implementation. We discuss recent contributions concerning shortest paths, minimum cost network flows, traffic equilibrium, vehicle routing, and network design and we enumerate several open research problems. Much of our discussion reflects an emerging theme in the analysis of transportation problems, the blending of ideas from transportation science, computer science, and operations research.

1. Introduction: Modeling and Implementation Issues

Transportation is vital to any society. It exerts great influence on the flow of goods, services, and information, on the location of homes and industry, on the use of recreational and cultural activities; in many ways, it does much to define the character of our lives.

Needless to say, planning for effective transportation is a complicated process requiring estimation of transportation needs, technological innovations, assessment of new and proposed investments, and efficient management of existing facilities. In most planning efforts, it is natural to view a transportation system as a transportation network with a number of nodes (representing street intersections, depots, ports, cities, and so on) and a number of links, arcs, or edges (e.g., streets, air and ship routes, or subway channels). Network models, hence, become the focal point for a great deal of analysis of transportation systems. The models may be normative, the optimization of system performance (usually with costs or travel times associated with the links) being the objective. Or, they may be predictive. How will users (individuals, private and public organizations) and the transportation industry itself respond to various policy alternatives? We consider both types of models in this paper.

Much of the recent research in network analysis for transportation planning has involved a blending of ideas from transportation science, operations research, and computer science. The interplay between modeling, algorithms, and their efficient computer implementation has become a dominant theme. To illustrate the nature of this research, and at the same time consider areas rich in applications, we have selected the following topics for discussion: shortest paths, minimum cost network flows, traffic equilibrium, vehicle routing, and network design.

We might note that our coverage of the first three of these topics, when contrasted with Potts and Oliver's highly-regarded book[105] published only a few years ago, is indicative of the current level of activity and recent progress in transportation planning and network analysis.

Transportation Models

The streets of a city form an obvious network where nodes (numbered for identification purposes) represent locations and intersections on roads, and links represent the roads themselves. We distinguish between one-way and two-way streets by using a directed link for the former and an undirected link (or

two directed links) for the latter. As an illustration of how a real transportation system may be abstracted by a network model, Figure 1 shows a segment of a fictitious city street network.

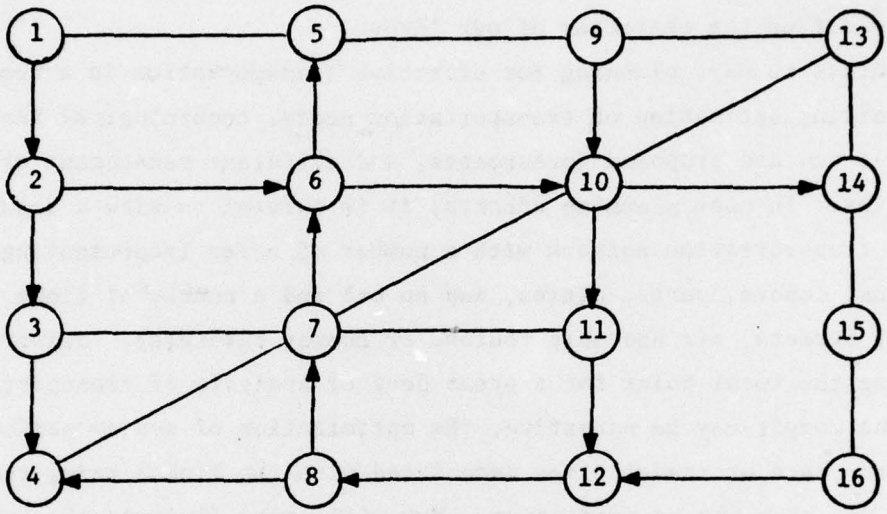


Figure 1. A City Street Network

In dealing with a network model of a real transportation system, transportation planners typically associate various parameters with the nodes and links. For example, each link of an urban road network may have values for the following items:

(i) number of traffic lanes,
(ii) road length,
(iii) average travel time,
(iv) average vehicle speeds,
(v) average daily traffic flow,
(vi) peak hour flows,
(vii) capacity
(viii) total monetary cost (including tolls).

These values are frequently combined in order to obtain a single measure of cost or distance on the link.

Different types of networks arise in other settings. What, for example, is the maximum income for an airline system? Given a number of possible nonstop services or flights, an associated expected income for each service, and

an overnight holding cost for an aircraft, what set of services should be flown for maximum system income (Simpson [116] describes this model and many others). Figure 2 provides a representation of such a problem as a time-space network. There are three geographical locations A, B, and C. The network consists of nodes which indicate both geographic location and time of day. Potential flights are shown by "service" arcs joining geographic locations at various times of the day; expected incomes are associated with these arcs. An arc from the end of the day to the beginning of the next day corresponds to holding an aircraft overnight. There are daily rental costs associated with these arcs. The problem is to find the maximum revenue route schedule subject to the capacity limitation that at most one plane flies any service arc.

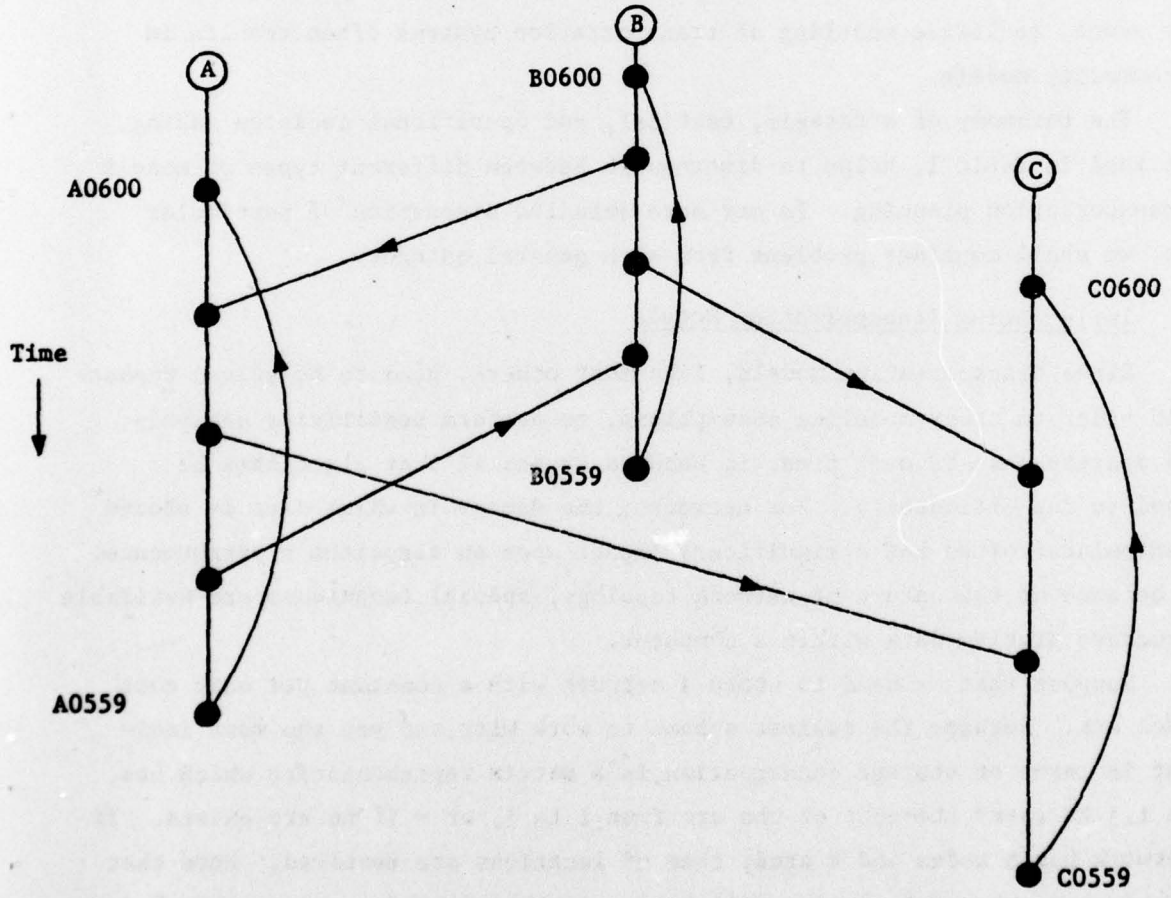


Figure 2. An Airline Schedule Map

Although one might usually associate vehicles with the examples of Figures 1 and 2, networks model other aspects of transportation planning as well, such as the flow of passengers, cargo, and vehicle crews. In fact, one of the most noteworthy features of transportation systems, and their representation as models, is that they usually involve several different commodities. In many instances commodities will be distinguished by their points of origin and destination. Passengers traveling from New York to Los Angeles are not indistinguishable from those traveling from Washington to San Francisco, even though they may share some of the same transportation facilities; otherwise, the model could route the New York passengers to San Francisco and those from Washington to Los Angeles. When passengers (or cargo or crews) constitute one type of commodity and vehicles another, the models are further complicated because the vehicle flows define possible routes and capacity limits for passenger travel. In any event, realistic modeling of transportation systems often results in multicommodity models.

The taxonomy of strategic, tactical, and operational decision making, as outlined in Table 1, helps to distinguish between different types of models for transportation planning. In our more detailed discussion of particular models, we shall consider problems from each general category.

Implementing Transportation Models

Since transportation models, like most others, need to be solved repeatedly in order to study modeling assumptions, to perform sensitivity analysis, and to address changes over time, it becomes essential that algorithms be designed to run efficiently. For networks, the manner in which data is stored and manipulated often has a significant impact upon an algorithm's performance. Also, because of the nature of network topology, special techniques are available to structure problem data within a computer.

Suppose that we need to store a network with a constant per unit cost for each arc. Perhaps the easiest scheme to work with, and yet the most inefficient in terms of storage conservation, is a matrix representation which has as the i,j th entry the cost of the arc from i to j, or ∞ if no arc exists. If the network has n nodes and E arcs, then n^2 locations are required. Note that for sparse networks most entries will be ∞. Another way of storing network data is known as the "ladder representation." For each arc we record its origin node, its destination node, and its cost. This approach calls for 3E locations.

Table 1. Taxonomy of Transportation Planning Models

Category	General Characterization	Most Frequently Used Optimization Models	Most Frequently Used Optimization Techniques	Some Examples
Strategic Planning	Economic Investment Decisions Long Time Horizon Highly Aggregate Data	(Mixed) Integer Programs Linear and Nonlinear Programs for Incremental Improvements to Existing Facilities	Branch and Bound Benders Decomposition Heuristics	Network Design [96] Fixed Charge Location Problems [112] Warehouse Location Models [60] P-Median and P-Center Location Models [25]
Tactical Planning	Resource Utilization Medium Time Horizon Aggregate Data	Linear and Nonlinear Programs	Specialized Implementations of the Simplex Method Out-of-Kilter Algorithm Frank-Wolfe and Other Primal NLP Algorithms	Routing Models: Transportation, Transshipment, and Multicommodity Flow [51] Traffic Equilibrium [47] Aggregate Inventory and Work Force Models [75]
Operational Planning	Execution Short Time Horizon (Possibly Real-Time) Detailed Data	Integer Programs Nonlinear Programs	Simulation Heuristics Dynamic Programming Optimal Control	Time-Table Scheduling [4], [110] Synchronization of Traffic Lights [54] Control of Corridor Traffic [80] Routing of Pick-up and Delivery Vehicles [17], [121] Management of Personal Rapid Transit (PRT) Systems [56]

A third representation, the "forward star representation," records the arcs ordered by origin node. An arc list contains for arc k, its destination node and its cost. An auxiliary node list records for node i, the first entry in the arc list originating from that node. The scheme requires n + 2E storage locations.

Now suppose the cost functions are of a more complex nature. For example, if the cost on an arc with flow x is given by a quadratic function $ax^2 + bx + c$, then we could either keep three cost matrices, or store three cost vectors A, B, and C using the ladder or forward star representations. These approaches would require $3n^2$, 5E, and n + 4E storage locations respectively. The storage schemes would be used in the same way to record other data, such as arc capacities.

Depending on the functional form of the cost functions, the sparsity of the network, and the amount of data manipulation required by an algorithm, the user must determine the best network representation for his particular application. We discuss this issue, and how it relates to implementing network algorithms, in subsequent sections of this paper.

For additional general information regarding transportation networks, the excellent surveys by Bradley [19] and Gazis [56] are recommended. Also, see Gartner et al. [53] and Steenbrink [118]. For an extensive bibliography on network optimization, see Golden and Magnanti [68].

2. Shortest Paths

Despite the number of papers on shortest path problems surveyed by Dreyfus [40] and later by Gilsinn and Witzgall [65], new insights regarding this class of problems continue to emerge. In the last few years, a number of algorithmic improvements have been reported which impact directly on transportation planning. In this section, we outline some of these recent contributions.

Overview

Shortest path problems are pervasive in transportation planning for several reasons. One of the primary objectives of any traveler (a passenger or a carrier) is to move from one point a to another point b, along a shortest, cheapest, or most comfortable path. Associating flow costs (distances or comfort factors) with arcs in a network, the traveler seeks the minimum cost path from a to b. In economic terms, there is a supply of one or more units at node a, a demand for these units at node b, and a link flow cost assigned to each link in the network.

Shortest path problems also arise in situations where this model, by itself, is not appropriate, such as when the route selected by one traveler effects the cost of routes taken by other travelers. In this paper, we discuss a number of important problems and techniques in network optimization relating to transportation. In some way, each problem relies or builds upon a shortest path algorithm. The minimum cost network flow problem is a generalization permitting supplies and demands for flow at various points in the network and flow capacities on the links. The network design problem introduces link construction possibilities. When urban transportation planners try to forecast traffic, the shortest path problem becomes an important subproblem. The vehicle routing problem requires a shortest path matrix as input. As these problems illustrate, a shortest path algorithm is at the core of many problems in transportation planning.

In terms of modeling transportation networks, it is important to realize that, in computing shortest paths, total travel time between points a and b depends not only on link travel times, but also on delays at intersections, often attributable to left hand turns. Network formulations model these situations by imposing turn penalties, that is, by associating costs or delays with turns at nodes. Turn prohibitions, which are enforced as policies in many transportation systems, can be regarded as turns with infinite penalties.

Several researchers have proposed algorithms for determining shortest routes in networks with turn penalties (see Potts and Oliver [105] and Kirby and Potts [88] for details). Although we do not pursue this topic here, we mention this issue because of its important modeling implications. An example other than a road network that might be modeled with turn penalties is a subway system with many different lines. Switching lines involves a delay and possibly a transfer charge.

Transportation planning is not the only setting in which shortest path problems are of interest. Similar applications arise in computer-communication studies. In addition, shortest path problems often become subproblems for more complex problems such as in group theoretic integer programming (see Shapiro [113], Chen and Zionts [24], Frieze [52], and Denardo and Fox [34]). In fact, computational studies of shortest path algorithms have inspired research in sorting, data structures, and list processing by operations researchers and computer scientists alike.

For a given network G = (N, A, D) with node set N, arc set A, and arc costs given by the matrix D = [d(i,j)], there are five shortest path problems of general interest.

(1) Find the shortest path from a specific origin s to a specific destination t;

(2) Find the shortest paths from a specific origin s to all other nodes;

(3) Find the shortest paths between all pairs of nodes;

(4) Find the shortest path between an origin-destination pair that passes through specified nodes;

(5) Find the second, third, and so on, shortest paths.

The distance entries d(i,j) can be positive, negative, or zero provided that there exists no cycle whose total cost is negative. If a negative cycle did exist, costs would be minimized by traversing it infinitely often.

Implementation Issues

Because shortest path problems are so central to transportation science, efficient implementation of computer codes for these problems often translates into substantial savings. At times, the efficiency of a code dictates the size of networks, and hence the detail of modeling, that can be analyzed. Furthermore, in real-time planning situations, fast computer codes become a necessity.

In the following discussion, we focus primarily on the second problem listed above which is, perhaps, the most common. We view this problem and Bellman's algorithm [12] for solving it as a vehicle for illustrating how rather minor changes in a

code's implementation can lead to significant reductions in computer running time. Both Bellman's algorithm [12], and a modification of it proposed by Pape [104] that we will consider, are classified as "label-correcting" procedures (see [65]), in the sense that tentative shortest path distances assigned to the nodes are revised until true shortest path distances are determined. We outline each procedure below:

BELLMAN'S ALGORITHM (also known as the Ford-Bellman-Moore Procedure [40])

Definitions

$\ell(v)$ is the length of the current "shortest path" from node s to node v.

$p(v)$ is the predecessor of v in the current "shortest path" to this node.

$d(i,k)$ is the length of arc $(i,k) \in A$.

Initialization

$$\ell(v) = \begin{cases} 0 & \text{if } v = s \\ \infty & \text{otherwise} \end{cases}$$

$p(v) = 0$ for all v.

node s is the first element on list T.

Basic Computation

Select the top element i from list T. For every node k such that $(i,k) \in A$, perform the following test:

If $\ell(i) + d(i,k) < \ell(k)$ then

(a) $\ell(k) = \ell(i) + d(i,k)$

$p(k) = i$, and

(b) place k at bottom of T, if it is not already on the list.

Reducing the List Size

Remove (or cross out) node i from the list. Terminate the procedure if the list T is now empty. Otherwise return to the basic computation.

PAPE'S ALGORITHM

This procedure is the same as the previous one except that we replace (b) of the basic computation step with:

(b') If k is already on list T, do not add it again.

If k has not yet been on the list, place it at the bottom of T.

If k has already been processed (that is, was on the list once before but is not currently), then enter k at the top of the list.

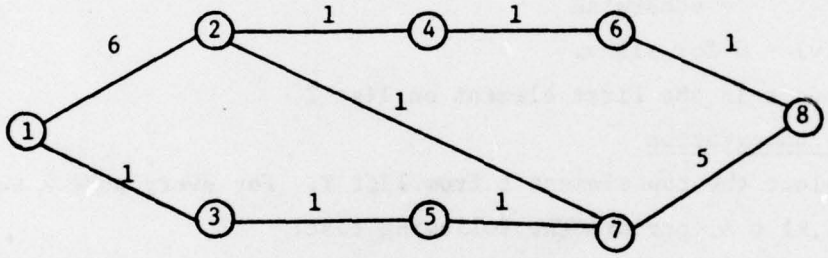
We point out that both algorithms are based on the following fundamental recursion

$$\ell(i) \leftarrow \min_j \{\ell(j) + d(j,i)\}$$

where the labels $\ell(v)$ are updated whenever a path of one additional arc has a smaller length than the previous best path. In addition, each algorithm requires on the order of n^3 additions and comparisons in the worst case, where n is the number of nodes in the network. There are other shortest path algorithms known as "label-setting" procedures (see [65]) that only require on the order of n^2 operations in the worst case.

The following example will illustrate the computational advantage of the second approach over the first. Furthermore, recent computational studies by Pape [104] and Klingman et al. [89] demonstrate that this approach seems to outperform other types of shortest path algorithms as well, including frequently advocated "label-setting" procedures, such as Dijkstra's algorithm[36].

Example 1. Find the shortest paths from node 1 to all other nodes in the undirected network below.



The reader is encouraged to determine the shortest paths by performing the steps indicated in Bellman's and Pape's algorithms. A crucial computational consideration in both instances is the length of list T. To be precise, the number of elements that have been placed on the list will determine the number of executions of the basic computational step, the most costly step in both procedures. With this in mind, let TB be the list of nodes for the Bellman algorithm and let TP be the list of nodes for the Pape algorithm. To avoid confusion, we performed the basic step in ascending order of k. In other words, we must consider arc (1, 2) before arc (1, 3), and so on. The lists are given below.

TB	TP
1	1
2	2
3	3
4	4
7	7
5	5
6	7
8	2
7	4
2	6
8	8
4	
6	
8	

In this case, Pape's algorithm requires almost 25% fewer repetitions of the basic computational step than does the original Bellman algorithm. In addition, this simple example is indicative of more general problems which Bellman's algorithm encounters quite frequently and which Pape's algorithm is capable of avoiding. Using Bellman's approach in example 1, node 2 receives an incorrect minimal distance label early in the procedure and seven other nodes are added to the list before node 2 receives its correct minimal label. Pape's approach adds only five other nodes before correcting the same initial error. In general, the great advantage of the new algorithm is that errors in minimal distance labels are corrected as soon as they are detected. This is accomplished by placing the node with the corrected label at the top rather than bottom of the list. Pape recommends a "deque" for the list T and he discusses its storage as well as computational savings (see [104] for details). A deque (or double ended queue) is a linear list in which all insertions and deletions are made at the ends of the list.

Problem 2 continues to generate research attention. Golden [66] has studied Problem 2 for Euclidean networks only. More recently, Denardo and Fox [33] have introduced a new family of shortest path algorithms based on buckets. A bucket is a list of nodes whose labels fall within a given range.

Let m (assumed positive) denote the length of the shortest arc in A. Then, define buckets of width m such that bucket p is a list of nodes i whose temporary labels v(i) fall (currently) in the interval.

$$mp \leq v(i) < m(p + 1), \quad p = 1, 2, \ldots$$

In Dijkstra's algorithm nodes are classified either as permanently or temporarily labeled. A permanently labeled node is one with a label which has been shown to be the true shortest path distance. At each iteration, the algorithm finds the node with the smallest temporary label defined by $v(j) = \min \{v(i) + d(i,j) : i \text{ is permanently labeled}\}$ and makes the label permanent. With buckets, we can replace this step with the determination of the lowest-numbered bucket p* that contains one or more temporary labels. Suppose at the end of an iteration that v(k) is the smallest temporary label. Then, by the very nature of the Dijkstra algorithm, all nodes i such that

$$v(k) \leq v(i) \leq v(k) + m$$

must be permanently labeled (see [33] for details). Since m is the length of the shortest arc, it is impossible for i to receive a label less than v(i) from node k or any temporarily labeled node. This observation can result in substantial computational savings.

We note that deGhellinck [32] has had encouraging preliminary computational experience imbedding the bucket approach to shortest paths within the out-of-kilter algorithm for solving transshipment problems.

The research that we have been discussing is primarily at the "implementation level" with the goal of developing faster and faster shortest path algorithms. Currently, problems with thousands of nodes are being solved in fractions of a second. Since these procedures are called upon routinely by transportation planners in so many applications, this development is worth following.

The other four shortest path problems mentioned earlier have also received attention in recent years. Hart et al. [73], Nemhauser [98], and Golden and Ball [67] discuss the application of a generalization of Dijkstra's algorithm to Problem 1. Floyd's algorithm [50] is still widely cited for Problem 3. As an alternative, we can repeat Pape's algorithm from each node. Dreyfus [40] has proposed an algorithm for solving Problem 4. Kershenbaum et al. [87] also solve this problem in the context of telephone network routing. Regarding Problem 5, Shier [114], [115] has developed an extremely effective procedure for finding the k shortest paths from a given node to all other nodes in a network. Dantzig et al. [30] have recently studied decomposition techniques for solving large scale shortest path problems.

We recommend Dreyfus [40], Gilsinn and Witzgall [65], and Christofides [25] as general sources of information and references on shortest path problems.

3. Transshipment Models

In making shortest path trip selections, travelers assume that their actions do not effect one another. Whenever transportation systems are operating near capacity, however, there are congestion delays and this assumption becomes untenable. In these situations, realistic models should account for interactions between the users. Another extension to the shortest path model is the classical transshipment problem in which goods are to be moved simultaneously from several sources to several destinations, for example, when empty rail freight cars are to be transported from their current rail yard locations to other yards where the cars are needed to provide hauling services [124]. In this section, we study some of these important extensions to the shortest path model, concentrating on recent algorithmic contributions for solution of the transshipment problem.

Modeling Considerations

To set notation, we cast the transshipment problem as follows:

$$\text{minimize} \sum_i \sum_j c_{ij} x_{ij}$$

$$\text{subject to} \sum_j x_{ij} - \sum_r x_{ri} = b_i \quad (i = 1, 2, \ldots, n)$$

$$0 \le x_{ij} \le u_{ij} \quad \text{all arcs } (i,j).$$

In this formulation, which models the flow of a homogeneous good, be it a class of passengers, freight, vehicles, or crew personnel, the decision variable x_{ij} is the flow of the good on arc (i,j), c_{ij} is a given per unit flow cost, and u_{ij} is a given upper bound (possibly $u_{ij} = +\infty$) for flow on arc (i,j). The quantity b_i is the known supply at node i, a negative value being interpreted as a demand. In practice, few of the possible arcs in this network model will correspond to links in the underlying transportation network, with three to five times as many links as nodes being typical. Accordingly, we assume that the summations and indexing in the model are restricted to links (i,j) and (r,i) of the physical network. This characteristic of network sparsity implies that a forward star representation of the data is an attractive storage scheme.

In the past few years, remarkable advances have been made on these problems. Using contemporary special-purpose network codes, it is now possible to obtain solutions up to two orders of magnitude faster than by solving these problems with general-purpose commercial linear programming packages. Solving the transshipment problem efficiently, like solving shortest path problems efficiently,

is important for two reasons. First, the transshipment problem realistically models a number of distribution and transportation decision-making situations. The empty freight car redistribution problem mentioned previously in this section and the aircraft scheduling model introduced in an earlier section are but two examples. Moreover, the repeated solution of this model is often embedded within procedures for solving more complex transportation problems.

As an illustration of this second point, consider a multi-fleet routing version of the aircraft scheduling model with, say, 707's, 727's, and 747's as airplane types in the fleet. The constraints of the transshipment model with superscripts k = 1, 2, or 3 differentiating between plane types on all variables x_{ij}^k and on all data b_i^k, c_{ij}^k, and u_{ij}^k, models the routing of each individual plane type. The multi-fleet model includes additional "bundle" constraints of the form

$$x_{ij}^1 + x_{ij}^2 + x_{ij}^3 \leq 1$$

on all service links (i,j), for example, a potential flight leg connecting Boston at 8 AM to Atlanta at noon. The bundle constraint ensures that only one aircraft type, if any, provides this service. In this example, we would require, as well, integral values for the decision variables x_{ij}^k.

The more general version of this multicommodity flow problem involves $K \geq 2$ commodity types, each subject to its own transshipment constraints. Bundle constraints imposed upon certain arcs of the network with bundle capacities, not necessarily equal to one, model interactions between the commodities. The model might be formulated as a linear program or might be formulated as an integer program and solved via branch and bound using linear programming. In either case, two different solution strategies, each of which decomposes the problem into a number of transshipment models, are candidates for solving the linear program. Price-directive decomposition places a value (or price) λ_{ij} on the bundle capacity of each arc (i,j) and "charges" for use of this capacity in a modified (Lagrangian) objective function

$$\sum_{k=1}^{K} \sum_i \sum_j (c_{ij}^k - \lambda_{ij}) x_{ij}^k$$

to be minimized. Resource-directive decomposition allocates the capacity K_{ij} of each bundle arc (i,j) among the commodities, i.e., imposes additional capacities y_{ij}^k on the flow variables such that $0 \leq x_{ij}^k \leq \min (y_{ij}^k, u_{ij}^k)$. Feasible allocations of the bundle capacities require that $\sum_{k=1}^{K} y_{ij}^k = K_{ij}$. Both algorithms operate by fixing values of the new variables λ_{ij} or y_{ij}^k and discarding the bundle constraints so that the problem then separates into K independent transshipment models, one

for each commodity type. Iteratively, the values of these variables are readjusted until an optimal solution to the problem is computed. The price-directive decomposition approach, implemented as Dantzig-Wolfe decomposition, has been very successful in several recent computational studies. Assad [6], [7], Kennington [84], [85], Kennington and Shalaby [86], and Swoveland [120] discuss further details of these algorithms, describe a number of applications, and report on computational experience.

An alternative approach to modeling of multicommodity flow problems accounts for interactions between the commodity types by incorporating congestion effects into the objective function. The transshipment constraints for each commodity are modeled as before, the bundle constraint is eliminated, and the objective function is replaced by

$$\text{Minimize } f(x),$$

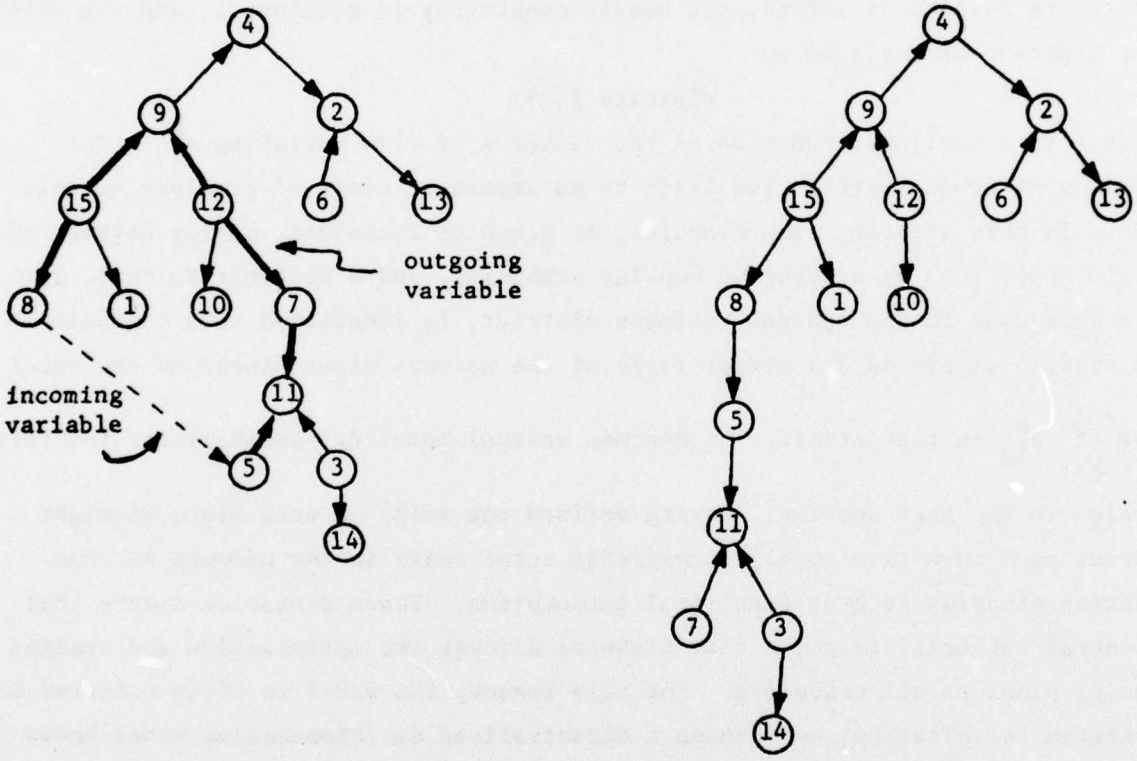
where f is a nonlinear function of the vector x of flow variables x_{ij}^k. The modeling of urban traffic flow leads to an important class of problems of this type. In this setting, each traveler, or group of travelers, moving between an origin node, such as a suburban housing community, and a destination node, such as a work zone in the central business district, is identified as a commodity. The traffic delays on any street (i,j) of the network might depend on the total flow $\sum_{k=1}^{K} x_{ij}^k$ on that street. We discuss several modeling possibilities for this problem in the next section. Having defined the delay on each link, we might choose, as a normative model, to minimize total delay in the network or some function of delay such as total fuel consumption. These scenarios assume that a central authority (e.g., a city planner) directs the optimization and assigns routing plans to all travelers. For this reason, the model is often referred to as system optimization; we discuss a decentralized decision-making model known as user optimization in the next section. As we shall see, user optimization frequently leads to the same type of multicommodity flow model.

We postpone discussing solution techniques for this nonlinear multicommodity flow problem until after we have introduced the user optimized problem and described urban traffic flow modeling in greater detail in the next section. At this point, we merely note that one algorithmic strategy is to linearize the objective function and repetitively solve transshipment problems, which at times, are simple shortest path problems.

Consequently, solving both the bundle and congestion forms of the multicommodity flow problem requires repetitive and, hence, efficient solution of the single commodity transshipment model--a topic that we consider next.

Solving the Transshipment Problem Efficiently

Because of the special network structure of the transshipment model, considerable streamlining is possible in implementing the simplex method to solve the problem. Figure 3 shows a typical iteration of the simplex method when applied to a 15 node transshipment model.



Adding Arc (8, 5) to the Basis

New Basis

Figure 3. A Simplex Iteration

The solid arcs in this figure, these arcs correspond to the variables x_{ij} in the linear programming basis, illustrate a fundamental and well-known property of this class of problems: every linear programming basis corresponds to a spanning tree in the underlying network. That is, when arc orientations are ignored, (i) the basic arcs contain no circuit, and (ii) any nonbasic arc forms a unique circuit with the basic arcs. The darkened arcs in Figure 3 are the basic arcs in the circuit formed by the nonbasic arc (8, 5).

In the simplex algorithm, the basis is updated from step to step by introducing a nonbasic arc to replace one of the basic arcs. This update requires:

(1) determining the circuit formed by the basis and the arc being introduced, which we call the *pivot circuit* (knowledge of the circuit and the problem data determines the arc to leave the basis), and

(2) recomputing the simplex multipliers, or node potentials, π_i, that satisfy

$$0 = c_{ij} - \pi_i + \pi_j$$

for every arc (i,j) in the new basis.

At each step, any nonbasic arc (i,j) with $\bar{c}_{ij} \equiv c_{ij} - \pi_i + \pi_j < 0$ becomes a candidate to enter the basis. The method of selecting from these candidate arcs, though something of an art, has a profound effect upon solution time. The most successful methods choose a subset of arcs, varying from step to step, and introduce the arc (p,q) whose reduced cost $\bar{c}_{ij}$ is minimal within the subset. Mulvey [97] and Bradley et al. [20] describe several mechanisms for implementing this strategy.

Efficient implementation of requirements (1) and (2) necessitates the careful storage and manipulation of data describing the current basis. Since these issues have been so successful in improving algorithmic performance, and since they illustrate so nicely the use of computer science techniques in transportation applications, we describe the implementation details more fully.

In Figure 3, we have arbitrarily "rooted" the basis at node 4. Conceptualizing a basis in this manner actually facilitates implementation. In order to determine the pivot circuit formed by nonbasic arc (p,q), we, first, may find the unique paths P_p and P_q in the basis connecting these nodes to the root of the tree. Those arcs lying in just one of these paths together with (p,q) form the pivot circuit. Here, it is convenient to record the predecessor of each node (the root has none), which is the first node encountered when traveling from this node to the root, i.e., the next higher node in the tree. Then, to determine the paths P_p and P_q we simply trace predecessors to the root.

Computing the paths P_p and P_q and merging them to find the pivot circuit is inefficient in most circumstances, but particularly when the circuit is small and lies deep in the tree so that the paths P_p and P_q are long and share many common arcs. An alternative is to move upward through the tree from nodes p and q, one step at a time, until their paths meet. There are several ways to implement this strategy. For example, we can store the depth of each node in the

tree; the depth of a node is the number of arcs in the path joining the node to the root. In Figure 3, nodes 8 and 5 have respective depths of 3 and 5. Because the paths P_p and P_q must meet at the same depth, we may find the pivot circuit as follows. Move from the deeper of the nodes p and q, say p, toward the root to a node that is as deep as node q. Then, move towards the root concurrently on both paths P_p and P_q until the point where the paths first meet. An alternative is to store the number of successors of each node (i.e., the number of nodes lying below it in the tree). We then move from the node with fewer successors (choose arbitrarily when ties arise) towards the root until we again encounter the same node on both paths P_p and P_q. Both methods have been implemented successfully.

Having found the pivot circuit and next determined the outgoing basic arc by one of these techniques, we must then compute the simplex multipliers for the new basis. Note that dropping the outgoing arc, arc (12, 7) in Figure 3, splits the current basic tree into two subtrees. If we hold the simplex multipliers for all nodes in one of these subtrees at their current values, then to achieve $c_{ij} - \pi_i + \pi_j = 0$ for all arcs in the new basis (and $c_{pq} - \pi_p + \pi_q$ in particular), the simplex multipliers for every node in the other subtree must change in magnitude by $|\bar{c}_{pq}|$. Thus, to complete this step, we need to enumerate all nodes in one of the subtrees. It is, of course, attractive to enumerate the nodes in the smaller of the subtrees, an identification that is simple to make when the number of successor nodes has been stored. Sequencing the nodes properly, and maintaining this information from one iteration to the next, greatly facilitates enumerating the nodes of a subtree. One possibility is a sequence, or traversal, that "walks" through the nodes of the tree, starting with the root, from top to bottom and left to right. For our example, this sequencing would read 4 - 9 - 15 - 8 - 1 - 12 - 10 - 7 - 11 - 5 - 3 - 14 - 2 - 6 - 13 before and 4 - 9 - 15 - 8 - 5 - 11 - 7 - 3 - 14 - 1 - 12 - 10 - 2 - 6 - 13 after the basis change. These traversals satisfy two conditions: (i) the predecessor of each node appears in the sequence before the node itself, and (ii) directly following each node, in sequence, are its successors in the tree (if there are any).

Now suppose that we wish to perform the basis change indicated in Figure 3. Deleting the outgoing arc (12, 7) creates two subtrees. To enumerate the nodes below node 7 in one subtree and update the simplex multipliers, we extract from the traversal the sub-sequence 7 - 11 - 5 - 3 - 14. Knowing the number of

successors of node 7 determines the length of this sub-sequence, i.e., that it terminates at node 14. This information can also be obtained from the predecessor data since 2 is the first node after node 7 in the sequence whose predecessor is not in the sub-sequence, or could be recorded directly--for each node, store the last node in the sequence that is one of its descendents. The choice among these options depends upon trade-offs between storage and computation time.

We have now seen how several types of data structures(predecessor, depth or number of successors, traversal) might reduce computation time for the simplex method. The efficiency of the algorithm as a whole, however, also requires efficient updating of these structures from step to step. The change in the tree is rather simple conceptually. "Holding" the tree, with the incoming arc attached, at its root, we "cut" the outgoing arc. The tree then "falls" into its new position (see Figure 3). Note that the subtree below the cut in our example appears in the new tree with the path from node 5 to the cut at node 7 reversed, but the rest of the subtree remains unchanged. Exploiting this observation helps in updating the data structures efficiently.

State-of-the-art papers by Barr et al. [10] and by Bradley et al. [20] describe thoroughly this updating process, as well as other details of the algorithm and its implementation. These papers and an earlier survey by Magnanti [94] cite and review a number of previous contributions. In a related development, Aashtiani and Magnanti [2] have used similar data structures to reduce the computation time of the out-of-kilter method for solving the transshipment problem.

Theoretical Bounds on Efficiency

The implementations just described have proven to be effective in solving numerous problems; they lead to computation times of about 8 seconds on a CDC 6600 computer for solving 1500 node, 4300-5700 arc problems. Nevertheless, Zadeh [126], [127] has constructed arbitrarily large examples requiring a number of iterations that is exponential in N the number of nodes. These examples show that solution times can become prohibative as the problem parameter N becomes large. Algorithms for network problems like the transshipment problem are said to be "good" if the solution time for any example is bounded by a polynomial in N. Several researchers (Edmonds and Karp [41], Dinic [37], Karzanov [83]) have proposed good algorithms for the transshipment problem, or special cases. One result of their effort is a novel algorithm for the maximal flow problem whose running time is bounded by an order N^3 polynomial. Even [44] reviews the algorithm in detail and Baratz[9] shows that this run time bound is best possible. This analysis invites further investigations into the complexity of the transshipment model.

4. Traffic Equilibrium

How can a subway system, an expanded major artery, a new bridge, one-way street assignments, priority lanes, and other policy alternatives available to urban planners help to alleviate congestion in our cities? More specifically, how would users respond to these alternatives? What demands would they impose upon public transportation facilities, what routes would they select in their travel by private vehicles, and what levels of congestion would the system experience. In this section, we consider models and algorithms for predicting such behavior.

Ingredients

The flow pattern of an urban transportation network depends, to a large extent, on relationships between demand and congestion:

(1) as the number of users of any link (arc) of the transportation network increases, the delay time (impedence) along that arc increases, and

(2) as the delay times* increase, the demands of users for travel decrease.

The models that we consider attempt to predict flow by determining when the demand "forces" and delay time "impedences" equilibrate.

In practice, the most successful applications of urban transportation modeling have been limited to the modeling of a single transport mode, namely private vehicles. In this case, the delay time t_a along an arc a is frequently modeled as

$$t_a(f) = t_0[1 + \alpha(f/c_a)^{\beta}] \qquad (1)$$

Here f is the total flow of vehicles on the arc, c_a is the steady state capacity of the arc, t_0 is the free flow time and α and β are constants; values of $\alpha = 0.15$ and $\beta = 4$ are typical of those used in practice. Branston [21] describes a number of alternate formulations for the link delay curve $t_a(f)$ and makes several suggestions concerning the proper use of these functions in practice.

A weakness of delay function (1) is that it does not account for the fact that the delay along a link is often a function of flows on other links in the network. For example, the delay along a link feeding into a busy intersection might depend upon flow on other links feeding that intersection. Furthermore, since two-way streets are modeled as two (directed) arcs with opposite orientations, the delay in one direction is often a function of the traffic in the other direction due, in part, to left hand turns.

---

* Any user cost may be used in place of delay time throughout this discussion.

The demand component of urban transportation modeling usually concentrates on origin and destination points, O-D pairs, for travel. Households, businesses, and other end points for travel are aggregated into zones that are represented by nodes in the transportation network. Other nodes in the network, such as intersections, are transshipment points for vehicle travel.

Most systems currently available for predicting urban traffic flow, such as the UMTA (Urban Mass Transit Authority) Transportation Planning System, generate demands by considering trip production and attraction factors such as income and parking availability in the zones, and travel time between the zones. Having fixed interzonal demands by this trip distribution phase, a trip assignment procedure, such as the equilibrium model that we discuss in the next subsection, predicts the route choice that users make in order to meet their travel demands.

Because the traffic patterns generated in this second phase may provide new estimates of travel times between the zones, adjustments to demands may be called for. The ultimate prediction of urban flow would be obtained by iterating between the trip distribution and trip assignment phases in some way, either formally or heuristically.

This iteration can be automated by using demand functions in the equilibrium model that depend upon travel times with respect to prevailing network congestion. We might model demand between each O-D pair i as a function $D_i(u_i)$, possibly linear, of the shortest travel time u_i between that pair. More generally, demand could be expressed as $D_i(u)$, a function of the vector u of shortest travel times u_j between all O-D pairs $j = 1, 2, \ldots, n$. This extended formulation permits broader modeling capabilities, such as incorporating destination choice. Suppose, for example, that O-D pairs 1 and 2 represent travel from a given zone to each of two shopping districts. Dial's [35] extended "logit model" with

$$D_1(u) = d\,\frac{r_1 e^{\theta u_1}}{r_1 e^{\theta u_1} + r_2 e^{\theta u_2}}\,,\quad D_2(u) = d\,\frac{r_2 e^{\theta u_2}}{r_1 e^{\theta u_1} + r_2 e^{\theta u_2}} \tag{2}$$

where d is the total number of shopping trips to be made and r_1 and r_2 are attraction factors for the two shopping districts, permits the traffic assignment procedure make destination choices between the shipping centers.

When time dependent demand models are used, the second phase procedure simultaneously determines traffic distribution and traffic assignment with factors such as zonal income and parking availability held fixed. These models are short range planning tools. Longer range analysis, for instance, studies on

the impact of new transportation facilities on urban development, would require data relating variations in the "fixed" factors to demand.

For further discussion of transportation demand models, the reader might consult Domenenich and McFadden's monograph [39]. Nguyen [101], [102] considers a problem related to our discussion, namely estimating O-D zonal trips from observed link flows.

An Equilibrium Model

In his seminal paper [122], Wardrop posited two principles for determining the distribution of traffic in an urban network. The first of these, which is the basis for most urban transportation planning models, is a fundamental behavioral assumption about user objectives in traveling between a given O-D pair:

> "the journey time on all routes actually used are equal, and less than those that would be experienced by a single vehicle on any unused route."

The following mathematical model of equilibrium captures the principle:

$$T_p(h) \geq u_i \qquad \text{all } p \in P_i \tag{3}$$

$$T_p(h) = u_i \qquad \text{if } h_p > 0,\ p \in P_i \tag{4}$$

$$\left.\begin{array}{l} \\ \\ \end{array}\right\} \quad (i = 1,2,\ldots,n)$$

$$\sum_{p \in P_i} h_p = D_i(u) \tag{5}$$

where $T_p(h) \equiv \sum_{a \in A} \{t_a(h)\text{: arc } a \text{ belongs to path } p\}$.

This model is usually referred to as a user equilibrium and contrasts with the system equilibrium model introduced in the last section which corresponds to Wardrop's second principle.

In the formulation (3)-(5), A denotes the arcs of the network, $i = 1,2,\ldots,n$ denotes the O-D pairs, and P_i denotes a set of paths joining O-D pair i. Usually P_i consists of all paths joining O-D pair i, although it may contain a subset of these paths (e.g., only those that the user perceives). The term h_p is the number of vehicles using path p; u_i, $D_i(u)$ and $t_a(h)$ are the shortest travel times, demands and link delays introduced previously. $T_p(h)$, the sum of travel times along all links a belonging to path p, gives the total travel time on path p.

Condition (3) states that the travel time on any path joining an O-D pair must be at least as large as the shortest travel time. Condition (4) states that a user travels (denoted by $h_p > 0$) only on paths giving the shortest travel time

between any O-D pair. Equality (5) implies that all demand for travel between any O-D pair is satisfied by flow along paths joining that O-D pair.

As Rosenthal [107] has pointed out, this formulation is, in a sense, a continuous approximation to Wardrop's principle, since in practice vehicles are indivisible and the path flow variables should be integral. Weintraub [123] has studied relationships between the continuous and integral formulations. Since the integer model has not been implemented, we shall confine our discussion to the continuous model.

Several features of this model are worth noting. First, we observe that the link delay function for each arc a is written in terms of the entire flow pattern in the network and not merely in terms of the total flow

$$f_a \equiv \Sigma\{h_p: \text{ arc a belongs to path p}\} \tag{6}$$

on that arc. Thus the model can, in principle, incorporate the link interactions mentioned in the previous section. Second, because each demand function $D_i(u)$ is expressed in terms of the shortest path distances between all O-D pairs, the model has the potential to provide for destination choice modeling and other extensions.

Finally, we should recognize that the formulation (3)-(5) encompasses multimodal distribution. To illustrate this point, let us consider modeling of buses and autos as two alternate means of transportation. We envisage two networks, one for each mode (possibly copies of the same road network) identifying a mode with its network. Each path set P_i then represents O-D transport by one mode. The demand and link delay function will embody interactions between the modes. In this simple setting, O-D indices 1 and 2 might correspond to bus and auto travel between the same physical O-D pair; the logit model (2) is one possibility for expressing the demands $D_1(u)$ and $D_2(u)$ for the two modes. The interpretation of (2) is much the same as before; d is the total demand between the O-D pair and r_1 and r_2 are attraction factors for the modes.

Recently, Florian [45] and Abdulaal and LeBlanc [3] have proposed two mode equilibrium models along these lines and suggested algorithms to compute a solution. Note that this type of model is capable of computing traffic distribution, modal split, and traffic assignment simultaneously, in contrast to the one pass sequential approach of the widely-used UMTA Transportation Planning system. For related work see Bruynooghe [22], Florian, Nguyen, and Ferland [49], Evans [43] and, particularly, Florian and Nguyen [48].

Although, as we have seen, the equilibrium formulation (3)-(5) permits richness in modeling, calibrating the link delay functions and demand functions and computing equilibria for the most general formulation remains an unattained objective. One comforting feature of this modeling approach is that very mild restrictions on the problem data ($t_a(0) > 0$, $t_a(h') \geq t_a(h)$ whenever $h' \geq h$, $t_a(h)$ continuous, and $D_i(u)$ continuous and bounded from above) guarantee that an equilibrum exists. Aashtiani [1] recently established this fact using results from nonlinear complementarity theory.

The use of the equilibrium model as a planning tool has, to date, been limited to single mode private vehicle applications in which (i) the volume delay on each link depends only on the total flow f_a on that link as, for example, in (1), (ii) the demand between each O-D pair depends solely upon the shortest travel time between that origin and destination, i.e., demand is given by $D_i(u_i)$, and (iii) $D_i(u_i)$ is a decreasing function. The key to analyzing this situation is an observation made by Beckman, McGuire, and Winston [11], that the Kuhn-Tucker conditions to the following optimization problem (in variables h_p and d_i)

$$\left.\begin{aligned} \text{Minimize} \quad & \sum_{a \in A} \int_0^{f_a} t_a(\tau)\, d\tau - \sum_{i=1}^{n} \int_0^{d_i} g_i(y)\, dy \\ \text{subject to} \quad & \sum_{p \in P_i} h_p = d_i \quad (i = 1, 2, \ldots, n) \\ & h_p \geq 0,\ d_i \geq 0 \quad p \in P_i,\ (i = 1, 2, \ldots, n) \end{aligned}\right\} \tag{7}$$

are equivalent to the equilibrium conditions (3)-(5) when the Kuhn-Tucker multiplier λ_i for the i^{th} equality constraint is identified with the shortest travel time u_i between O-D pair i, if $d_i > 0$. In this optimization model $g_i(y) \equiv D_i^{-1}(y)$ is the inverse of the demand function (the model includes the special, but important, case of constant demands $D_i(u_i)$ by setting $g_i(y) \equiv 0$).

The importance of modeling the equilibrium problem as an equivalent minimization problem is that the objective function of (7) is convex whenever $t_a(\tau)$ and $g_i(y)$ fulfill the practical assumptions of being, respectively, nonincreasing and nondecreasing. Consequently, methods from convex programming can be applied computationally.

Computing An Equilibria

Of the several proposals that have been made for computing equilibria by solving (7), see Nguyen [99], the most widely used is the Frank-Wolfe algorithm.

This method solves nonlinear programs with linear constraints, i.e., $\min \{f(x): Ax = b, x \geq 0\}$, by repeated linearization of the objective function. Given any feasible solution x^j to the problem, the method finds a solution y to the linearized problem

$$\min \{\nabla f(x^j) \cdot y: Ay = b, y \geq 0\} \tag{8}$$

where $\nabla f(x^j)$ is the gradient of f evaluated at x^j. It then solves the one-dimensional search problem of minimizing f in the line segment joining x^j and y, obtaining a new solution x^{j+1}. The method iterates over $j = 0, 1, 2, \ldots$ starting with an arbitrary initial feasible solution x^0.

This algorithm is particularly well suited for solving problem (7). Consider, first, the fixed demand model in which d_i is a constant and $g_i(y) \equiv 0$ for $i = 1, 2, \ldots, n$. Any linear objective function $\sum_{i=1}^{n} \sum_{p \in P_i} C_p h_p$ is minimized subject to the constraints of (7) by setting $h_{q_i} = d_i$ where $q_i \in P_i$ satisfies $C_{q_i} = \min \{C_p : p \in P_i\}$; that is, the demand is met by any minimum cost path. Some algebraic manipulations reveal that the coefficient C_p of h_p obtained by linearizing (7) about any vector (h_p^j) of given path flows is simply the sum of $t_a(f_a^j)$ along arcs a belonging to path p. Consequently, the linearized problem (8) reduces to a sequence of shortest path problems, one for each O-D pair, with the prevailing link delays as arc costs. These shortest path problems can be solved efficiently by the techniques described in an earlier section of this paper.

Two points about the algorithm are worth noting. First, there is no need to enumerate all paths in each path set P_i prior to the analysis. The algorithm generates them as needed. Second, only the total flow f_a^j on each arc needs to be maintained from step to step. Once the one-dimensional line search has been performed at each step, the shortest path solutions can be discarded. Exploiting this fact leads to substantial reductions in storage requirements.

The variable demand version of (7) is solved in much the same way. When linearized, the objective function coefficient of d_i for problem (7) becomes $u_i \equiv g_i(d_i^j)$. The solution to the subproblem (8) then depends upon both the values of C_{q_i} as defined above, and the current shortest path distances u_i. A solution is:

$$h_p = 0 \qquad \text{if } p \neq q_i$$

$$h_{q_i} = d_i = \begin{cases} 0 & \text{if } C_{q_i} > u_i \\ d_i^j & \text{if } C_{q_i} = u_i \\ b_i & \text{if } C_{q_i} < u_i \end{cases}$$

where b_i is any known upper bound on the demand between O-D pair i. Nguyen [100] describes further details about this algorithm and discusses other methods for solving for an equilibrium with variable demands.

Several researchers have contributed ideas related to this algorithm. The excellent survey [47] contains additional references and provides historical perspective concerning this and other algorithms for solving the minimization problem (7). We should emphasize that problem (7) is just one manifestation of the congestion formulation of the multicommodity flow problems discussed in the last section. Any algorithm for solving (7) usually applies to this broad generic set of models.

An alternative to casting the equilibrium problem in equivalent convex minimization form (7) is to view the model as a nonlinear complementarity problem. The model then can be studied from the viewpoint of this theory (Aashtiani [1], Hall [72]) or the viewpoint of fixed point theory (Kuhn [91], Kuhn and Cullum [92]). This approach has the advantage of applying to the general equilibrium formulation, but the disadvantage, to date, of requiring much greater computer time and storage than the minimization approach.

Computational Experience

To test the validity of equilibrium modeling as a predictive tool, Florian and Nguyen [46] applied model (7) to data from the city of Winnipeg. They assumed fixed travel demands, as generated from a previous study, and used an alternate to (1) for modeling link delays. The model predicted flow on high volume links quite well, but did not perform as well on links with observed volumes in the range of 0-300 vehicles per hour. Their findings show, as might be expected, that the predictions of route travel times were better than those of link travel times. They concluded that "the results are encouraging and demonstrate the suitability of the method for planning purposes."

In this study the Frank-Wolfe algorithm, equipped with a Dijkstra-type shortest path routine, required 15-18 iterations and about 700 CPU seconds on a CDC Cyber 74. The convex simplex method solved the same problem in about 500 CPU seconds, but required more storage. The network contained 1319 links.

In another study, Hern [78] considered a 9386 link, 3027 node network of Washington, D.C. The Frank-Wolfe algorithm, as implemented in the TRAFFIC computer code, required 221 CPU seconds per iteration on an IBM 360/91.

5. Vehicle Routing

Like many operational issues in transportation planning, the vehicle routing problem is encountered routinely and repeatedly in business and industry. The basic problem is one of designing a set of vehicle routes of minimal total distance leaving from, and eventually returning to, a central depot, which satisfies capacity constraints and meets customer demands. Demands occur at points or nodes in the transportation network and may be deterministic or probabilistic in nature. Generally, there are enormous amounts of detailed data and a far larger number of feasible sets of routes to consider. As a result, only small problems can be solved for optimal solutions; otherwise, we must reconcile ourselves to the fact that heuristic solutions (hopefully near-optimal) must suffice. In addition, there are a host of inter-related aspects of the vehicle routing problem including the number and location of depots and demand points, the capacity of vehicles and makeup of fleet, frequency of service, and other geographical considerations. The computational complexity of this problem and the fact that this type of problem is often solved every day underscores the need for powerful and efficient solution techniques.

Deterministic Setting

First, we focus on the case where demands are deterministic. Examples of this problem include municipal waste collection [13], fuel oil delivery [55], newspaper distribution [69], and routing of school buses [14]. Notice that in some examples pick-ups are made; in others deliveries are made. As long as only one of the two operations is performed throughout, the distinction is not important.

There are many heuristic techniques which have been proposed for this class of problems. We concentrate, in this section, on one approach which has been successful in solving large problems (more than 100 nodes). Gillette and Miller [64] and Orloff [103] discuss alternative heuristic strategies.

The algorithm we describe is an efficient implementation of the Clarke-Wright savings method [27]. Suppose we let node 1 denote the central depot and $d(i,j)$ be the distance from node i to node j. If every two demand points i and j are supplied individually by two vehicles from the central depot, then total distance traveled is $2\,d(1,i) + 2\,d(1,j)$. However, if both points are served by a single vehicle then the combined route results in a savings in travel distance of

$$\{2\,d(1,i) + 2\,d(1,j)\} - \{d(1,i) + d(1,j)\}$$
$$= d(1,i) + d(1,j) - d(i,j).$$

A similar savings occurs whenever the endpoints of two vehicle routes are joined to form a single route. In the Clarke-Wright algorithm, we proceed as follows:

Step 1. Evaluate all potential savings

$$S(i,j) = d(1,i) + d(1,j) - d(i,j) \quad \text{for } i,j \neq 1.$$

Step 2. Order all feasible savings from largest to smallest.

Step 3. Select the node pair (i,j) with the greatest positive feasible savings. Link nodes i and j on a single tour.

Step 4. Eliminate infeasible savings and return to step 2, until there are no remaining positive feasible savings.

When we say a savings S(i,j) is feasible we mean that linking nodes i and j does not cause the violation of any constraint (tour integrity, vehicle capacity, maximum route time, and so forth). After each linking of nodes, a number of savings become infeasible (see [69] for details). For example, an intermediate tour 1-2-i-3-1 implies that, for all k, savings S(i,k) are infeasible since otherwise we would not preserve the tour.

This algorithm can be coded efficiently by taking advantage of two important observations:

1. Step 1 can be very costly both computationally and in terms of storage requirements. For instance, a 600-node problem requires 360,000 storage locations for inputs (distances d(i,j) above the diagonal and savings S(i,j) below the diagonal for an undirected network with symmetric distances).

2. At each step of the algorithm, we must determine the maximum feasible savings.

These observations can be exploited by using special data structures and list processing techniques. First, rather than consider an entire matrix of pairwise linkings, we can focus on the most promising linkings only. Instead of storing the network topology in a matrix, we would record for each potential arc its origin node, its destination node, and its length, i.e., use a ladder representation. From this information we then calculate the savings. We restrict the entries in this list to reduce the number of savings considered to under 10,000 for a 600-node problem. One procedure for accomplishing this reduction is to consider linking node i to any node j within a

distance of r units of i. Golden, Magnanti, and Nguyen [69] document an alternative approach involving a rectangular grid. An extremely convenient and efficient method for finding the best feasible savings is to partially order the savings in a heap structure and update the structure from step to step (see [69]). These ideas lead to an implementation of the Clarke-Wright algorithm which is between one and two orders of magnitude faster than the traditional implementation.

Stochastic Setting

Now, we consider the more complex problem of constructing a fixed set of routes when demands are probabilistic. Such a problem would arise when daily deliveries of fuel oil are being made to automotive service stations and, although each route is fixed in advance, the demand on a particular day is stochastic. For simplicity, let us assume that the demand at each node i, denoted by d_i, can be modeled by a Poisson distribution with mean λ_i. The discussion here follows Stewart [119] and Golden and Stewart [70].

We say that a primary error has occurred if a vehicle cannot satisfy the demands of the customers on the route to which it has been assigned. This situation has various penalty costs associated with it. Clearly, one objective is to minimize the probability of a primary error. The stochastic vehicle routing problem can then be formulated as determining a fixed set of routes to:

Minimize (1) expected total travel distance
subject to (2) meeting customer demands;
(3) not exceeding vehicle capacity;
(4) Prob {primary error on a route} $\leq \alpha$.

We can solve the problem heuristically in such a way that we take advantage of the efficient Clarke-Wright implementation discussed in connection with deterministic demands. Suppose a route contains nodes $n_1, n_2, \ldots, n_k$ and has total demand and total expected demand

$$x = d_{n_1} + d_{n_2} + \ldots + d_{n_k}$$

$$E(x) = \lambda_{n_1} + \lambda_{n_2} + \ldots + \lambda_{n_k}, \text{ respectively.}$$

Then, by appealing to the Central Limit Theorem (see [70] for details) we can approximate the Poisson distribution for total demand by a Normal distribution with

$$\mu = \lambda_{n_1} + \lambda_{n_2} + \ldots + \lambda_{n_k} \quad \text{and}$$

$$\sigma = \sqrt{\mu}\,.$$

If we assume that all vehicles have the same functional capacity c, then the probability of a primary error is given by

$$\text{Prob } (x \geq c) = \text{Prob } \{z \geq \frac{c - \mu}{\sqrt{\mu}}\}$$

using the Normal approximation where z is a unit normal variate.

Our solution strategy will be to replace the functional capacity of a vehicle with a reduced "artificial" capacity and to replace the stochastic demand at each node with a deterministic "artificial" demand in such a way that the deterministic model may be used.

Let $\bar{\mu}$ denote the artificial capacity of a vehicle and λ_i the artificial demand at node i. If vehicles are loaded to their artificial capacities, then, solving

$$\text{Prob } \{z \geq \frac{c - \bar{\mu}}{\sqrt{\bar{\mu}}}\} = \alpha\,,$$

we obtain, after some algebra, the value of $\bar{\mu}$ which insures that constraint (4) is satisfied. That is,

$$\bar{\mu} = \frac{2c + z_{1-\alpha}^2 - \sqrt{z_{1-\alpha}^4 + 4c\, z_{1-\alpha}^2}}{2}\,.$$

where $z_{1-\alpha}$ is defined by Prob $\{z \leq z_{1-\alpha}\} = 1 - \alpha$. For instance, if $c = 100$ and $\alpha = .10$, then $z_{1-\alpha} = 1.28$ and $\bar{\mu} = 87.9$. Using an integral artificial capacity of 87 units, this gives a safety stock of 13 units as a cushion against the occurrence of primary errors. Fixed routes are constructed on the basis of the artificial capacity and artificial demands, as in the deterministic case. Sensitivity analysis is recommended for differing values of α. Golden and Stewart [70] have implemented this solution strategy and presented computational results. We point out that the procedure applies to many probability distributions other than the Poisson.

A potential transportation application involves the design of an effective subscription bus service for large employment centers. A reliable daily bus service would make pickups at pre-specified bus stops each morning, travel to the employment center, and make drops at the bus stops in the evening. There might be a fixed monthly fee for service. Since a subscriber need not show on a particular day because of illness, vacation, or special plans, we have a probabilistic vehicle routing problem.

6. Network Design

There are a number of options in the design of a transportation system, ranging from operational alternatives such as one-way street assignments to the physical improvement of existing facilities or construction of new facilities. We shall view any of these options as a network synthesis and refer to any alternative in terms of a network construction.

A general taxonomy of design problems divides into (i) arc construction in which certain arcs (e.g., roadways or railbeds) are added to the network, or not, and (ii) facility location models in which nodes representing warehouses, depots and the like are "opened," or not. In each case, the objective is to determine economic tradeoffs between the cost of construction and the savings in routing cost that it provides.

After briefly reviewing several models for assessing these tradeoffs, we consider, in this section, recent algorithmic advances for this class of problems. Our discussion focuses on three different approaches--branch and bound, Benders decomposition, and heuristic procedures.

Design Models

For $k = 1, 2, \ldots, K$ let r_k denote the flow requirement between nodes s_k and t_k of a given network; let c_{ij}^k denote the per unit routing cost on arc (i,j) for "commodity k" goods, and let f_{ij} denote the fixed cost of constructing arc (i,j). Then, a rather general network design model is:

$$\text{Minimize} \sum_k \sum_i \sum_j c_{ij}^k x_{ij}^k + \sum_i \sum_j f_{ij} y_{ij}$$

subject to

$$\sum_j x_{ij}^k - \sum_r x_{ri}^k = \begin{cases} r_k & \text{if } i = s_k \\ -r_k & \text{if } i = t_k \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

$$\sum_k x_{ij}^k \leq K_{ij} y_{ij} \tag{10}$$

$$\sum_i \sum_j e_{ij} y_{ij} \leq B \tag{11}$$

$$(x,y) \in S \tag{12}$$

$$x_{ij} \geq 0 \qquad \text{all } i \text{ and } j$$

$$y_{ij} = 0 \text{ or } 1 \quad \text{all } i \text{ and } j.$$

In this formulation, x_{ij}^k is the flow on arc (i,j) of goods being transported from node s_k to node t_k. The binary variables y_{ij} indicate whether or not an arc is constructed. Indexing conventions for this model are similar to those we have used for the transshipment model in section 3.

Equations (9) are the usual transshipment constraints. The "bundle" inequalities (10) specify that the total flow on arc (i,j) must be zero if that arc is not constructed (i.e., $y_{ij} = 0$) and cannot exceed the capacity K_{ij} of the arc if it is constructed. Inequality (11) is a budget constraint stating that total construction cost is limited by a budget B. In this expression, e_{ij} is a cost function, which need not equal f_{ij}, related to the building of arc (i,j). The set S encompasses further side constraints, possibly expressed as linear inequalities in the vectors $x = (x_{ij})$ and/or $y = (y_{ij})$. These might include precedence relations (e.g., construct arc (i,j) only if arc (i',j') is constructed), multiple choice relations (e.g., choose at most (at least, exactly) two of some subset of arcs), limitations on resources shared by several arcs, or prior specifications of arcs already constructed such as $y_{ij} = 1$. When prior specifications have been made, the model is often referred to as a network improvement model.

Most papers written about network design consider specializations of this model. We shall use the following terminology for these problem variants:

Uncapacitated Design--every $K_{ij} \geq \sum_k r_k$ so that constraints (10) impose no (capacity) restrictions on flow when $y_{ij} = 1$.

Fixed Charge Design--the budget constraint is eliminated: tradeoffs between fixed charge and routing costs are investigated.

Budget Design--the fixed charge term $\sum\sum f_{ij} y_{ij}$ is eliminated. Optimal routing is sought within a fixed construction budget.

We should note that although our formulation includes only one construction level K_{ij} for each arc, multiple capacity levels can be modeled by parallel links. Also, as an alternative to the bundle constraints (11), we might incorporate nonlinear congestion costs in the objective function. This type of formulation is attractive for continuous models where incremental improvements are being made to an existing network. Dantzig et al. [29], [30] and Harvey and Robinson [74] have considered such a model for budget design. They apply a Lagrange multiplier to the budget constraint, use the Frank-Wolfe algorithm (see section 3) for any fixed multiplier value, and iterate to find the optimal value for the multiplier.

A facility location model similar to this formulation of the network design problem is:

$$\text{Minimize} \sum_i \sum_j c_{ij} x_{ij} + \sum_i f_i y_i$$

$$\text{subject to} \quad \sum_i x_{ij} = d_j \tag{13}$$

$$\sum_j x_{ij} \leq K_i y_i \tag{14}$$

$$\sum_i e_i y_i \leq B \tag{15}$$

$$(x,y) \in S$$

$$x_{ij} \geq 0 \quad \text{all } i \text{ and } j$$

$$y_i = 0 \text{ or } 1.$$

In this case, the underlying network is bipartite. There are m possible locations for (production) facilities. The variable y_i equals 1 if site i is selected and is zero otherwise. K_i denotes the capacity of a site if it is selected, d_j denotes the demand at destination node j and f_i is a fixed cost for selecting site i. As before, (15) represents a budget constraint and the set S incorporates various side conditions. When the fixed charge term of the objective function is omitted, each $e_i = 1$, and $B = P$ is a limit on the number of facilities that can be selected, the model is called a P-median problem.

As demonstrated by Wong [125], the location model can be cast as a network design problem by adding an artificial node and an arc joining this node to every source node i. Let r_j denote the flow requirement from this new node to destination j and associate f_i, K_i, and e_i with the arc joining the new node with source node i.

Johnson et al. [81] have shown that the network design budget problem, even with each $e_{ij} = 1$, is NP-complete. That is, it can be solved by an algorithm that is polynomial in problem input (number of nodes, size of budget) if and only if any of a number of other notoriously difficult problems (including the traveling salesman problem, and the multicommodity flow problem in integers) can be solved similarly. This result provides some theoretical insight concerning the difficulty of design problems.

Branch and Bound

Several researchers have proposed branch and bound algorithms for solving network design and facility location models (Boyce et al. [18], Christofides and Brooker [26], Dionne and Florian [38], Leblanc [93] Scott [111] Steenbrink [118], Efroymson and Ray [42], Davis and Ray [31], Hoang [79] and Geoffrion and McBride [63], among others). To illustrate the nature of this work, we describe two of the more recent contributions from this list.

Several years ago, Hoang [79] suggested an enumerative algorithm for budget design problems with unit flow requirements between every pair of nodes. In this instance, the optimal routing, given any network configuration, is via shortest distance paths joining each origin-destination pair. Consequently, the objective function cost, denoted F(y), is determined completely by the network configuration, i.e., by the choice of 0-1 values for the components y_{ij} of the vector y. At any node P in the branch and bound enumeration tree, certain arcs A^F are fixed as constructed (either $\hat{y}_{ij} = 1$ or $\hat{y}_{ij} = 0$).

As a bounding mechanism for his algorithm, Hoang noted that any solution y with the arcs in A^F fixed at these same values satisfies the inequality

$$F(y) \geq F(y^P) + \sum_{(i,j) \notin A^F} \bar{y}_{ij} \, I_{ij}(y^P) \qquad (16)$$

where $\bar{y}_{ij} = 1 - y_{ij}$. In this expression, y^P denotes a solution with $y_{ij} = 1$ for every arc not in A^F and $y_{ij} = \hat{y}_{ij}$ for every arc $(i,j) \in A^F$. Thus $F(y^P)$ is the best possible shortest route solution with the given fixed values of arcs in A^F. $I_{ij}(y^P)$ denotes the increment to the shortest route cost from node i to node j when arc (i,j) is deleted from the network defined by y^P.

Expression (16) has the following interpretation. If arc (i,j) is deleted (set $y_{ij} = 0$ and $\bar{y}_{ij} = 1$) from the network defined by y^P, then the cost for shipping the unit of demand between these nodes must increase by at least $I_{ij}(y^P)$ in the solution y. It might increase by more because other arcs are being deleted as well. Hence, the right-hand side of (16) is a lower bound on the routing cost F(y) of solution y.

To compute lower bounds quickly, Hoang suggests relaxing the integer requirement on y_{ij} and minimizing the right-hand side of (16) subject to the budgetary constraint $\sum_i \sum_j e_{ij} y_{ij} \leq B$ and $0 \leq y_{ij} \leq 1$, $y_{ij} = \hat{y}_{ij}$ for arcs $(i,j) \in A^F$. A solution y^* to this continuous knapsack problem has at most one fractional component and, in light of (16), gives a lower bound $F(y) \geq F(y^P) + \sum_{(i,j) \notin A^F} \bar{y}^*_{ij} I_{ij}(y^P)$ that applies to every node below node P in the branch and bound enumeration tree. Using this lower bound as a fathoming mechanism and branching from node P on (i.e., next fixing) the fractionally valued variable in the continuous knapsack solution, Hoang implements his algorithm in the framework of a straightforward branch and bound algorithm.

Dionne and Florian [38] have noted several ways to improve this algorithm. First, in computing $I_{ij}(y^P)$ it is not necessary to resolve from scratch for shortest paths between all nodes. Specialized algorithms are available for recomputing shortest path distances when one arc has been deleted from a network. Second, they suggest branching on the variable y_{ij}, $(i,j) \notin A^F$ with highest incremental improvement per unit of budget, i.e., that arc (i,j) maximizing $I_{ij}(y^P)/e_{ij}$. These modifications lead to marked improvements in the algorithm. A typical example with a 65% budget level, i.e., $B = 0.65 \sum_i \sum_j e_{ij}$, with 20 nodes, and with 30 arcs, requires 20 seconds to solve on a CDC Cyber 74 computer with their algorithm, while Hoang's algorithm, after 500 seconds, is not able to determine that the current best solution is optimal. The authors note, however, that this branch and bound approach is probably limited to medium-sized networks like this, and that computation time seems to grow exponentially with a decrease in budget level. For example, the algorithm requires 288 seconds to solve the same problem with a 50% budget level.

In another development Geoffrion and McBride [63] consider Lagrangean relaxation embedded within a branch and bound approach to the fixed charge facility location problem. To adhere to their notation, let us assume that $d_j > 0$ for all j and let us substitute $z_{ij} \equiv x_{ij}/d_j$ in our formulation. They assume that the side conditions of S are expressed as a system of linear inequalities $Ay + Bz \leq b$ and attach a vector of Lagrange multipliers μ to these constraints and a vector of Lagrange multipliers λ to the constraint (13) to form a Lagrangean relaxation:

$$L(\lambda,\mu) = \underset{y,z}{\text{Minimize}} \sum_i \sum_j c'_{ij} z_{ij} + \sum_i f_i y_i + \sum_j \lambda_j (\sum_i z_{ij} - 1) + \mu(b - Ay - Bz)$$

subject to:

$$\sum_j d_j z_{ij} \le K_i y_i \quad \text{all } i \tag{17}$$

$$0 \le z_{ij} \le 1 \quad \text{all } i \text{ and } j \tag{18}$$

$$y_i = 0 \text{ or } 1 \quad \text{all } i \tag{19}$$

Here, $c'_{ij} = c_{ij} d_z$.

There are several reasons for considering this type of relaxation. The well-known "weak duality" property of Lagrangean duality demonstrates that $L(\lambda,\mu)$ is less than or equal to the optimal value v of the location problem for any value of λ and any $\mu \ge 0$. Thus, the Lagrangean relaxation provides a lower bound that can be utilized as a bounding mechanism in branch and bound. This particular Lagrangean relaxation is attractive because it is so easy to solve. It separates into independent subproblems, one for each i. For each i, either $y_i = 0$ and $z_{ij} = 0$ for all j or, $y_i = 1$ and the computation of optimal z_{ij} reduces to a continuous knapsack problem. Moreover, the value $L(\lambda,\mu)$ of the Lagrangean relaxation when λ and μ are set equal to optimal dual values of the linear programming relaxation provides a sharper (larger) lower bound on v than does the optimal value of the linear programming relaxation.

Computing the tightest Lagrangean lower bound, i.e., finding $L \equiv \max_{\lambda,\mu}\{L(\lambda,\mu) : \mu \ge 0\}$ yields insight into the structure of location models. Due to the equivalence between dualization and convexification (see Geoffrion [57] and Magnanti et al. [95]), L also equals the optimal objective value of the problem:

$$\underset{z,y}{\text{Minimize}} \sum_i \sum_j c_{ij} z_{ij} + \sum_i f_i y_i$$

subject to

$$\sum_i z_{ij} = 1 \quad \text{all } j$$

$$A y + B z \le b$$

and $(x,y) \in$ Convex Hull of solutions to (17), (18) and (19).

As Geoffrion and McBride point out, the convex hull constraints of this formulation are equivalent to the linear inequalities (17), (18), and $0 \le y_i \le 1$, together with the inequalities $z_{ij} \le y_i$ for all i and j. This result gives some theoretical justification for appending the constraints $z_{ij} \le y_i$ for all i and j to the location model. Moreover, Geoffrion and McBride conjecture that "linear

programming technology will advance to the point" that solving for L in this linear inequality form may be "preferable to involving Lagrangean relaxation."

Computational experience with problems ranging from 7 possible facilities and 122 transportation links to 25 possible facilities and 1250 links indicates that Lagrangean relaxation provides much tighter lower bounds on v than does linear programming relaxation. A branch and bound algorithm equipped with Lagrangean relaxation solved these problems in from 3.4 to 112.9 seconds on an IBM 370/158 computer compared with a range of from 6.8 to greater than 300 seconds for a branch and bound algorithm equipped with conventional Driebeek-Tomlin penalties.

Benders Decomposition

When applied to network design problems, Benders decomposition proceeds iteratively by choosing a tentative network configuration (i.e., setting values for the integer variables y_{ij}), solving for the optimal routing on this network, and using the solution to the routing problem in order to redefine the network configuration. Figure 4 illustrates this last step for an uncapacitated fixed cost design problem in which one unit of a good is to be sent from node 1 to node 6. In this instance, the routing problem reduces to a shortest path computation between nodes 1 and 6.

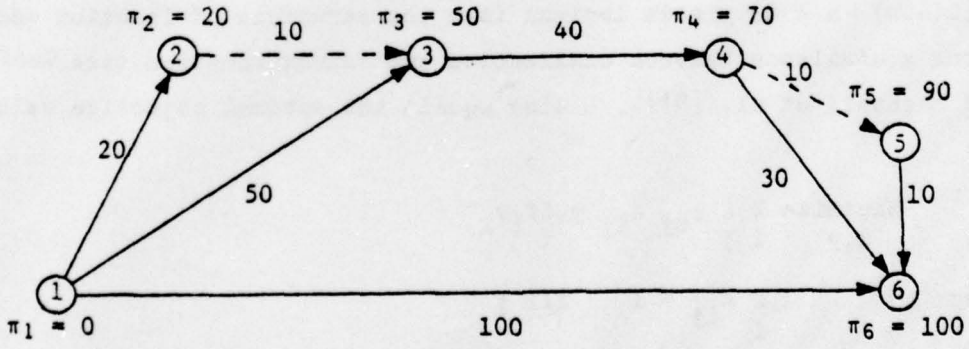


Figure 4. A Step of Benders Decomposition

The dark arcs in the figure are members of the current network configuration; the dashed arcs are candidates for inclusion in the optimal design. With respect to the routing costs shown next to each arc, the node numbers π_j are optimal dual variables for the linear programming routing problem. The dual variables indicate that introducing arc (2, 3) into the current design reduces

the routing cost from node 1 to node 3 from $\pi_3 = 50$ to $\pi_2 + C_{23} = 20 + 10$ for a savings of $50 - 30 = 20$ units. Similarly, introducing arc (4, 5) reduces the routing cost from node 4 to node 6 by $\pi_5 - (\pi_4 + C_{45}) = 90 - (70 + 10) = 10$ units. Since either of these arcs might, but need not, become part of the shortest route path from node 1 to node 6 in the optimal network design, the optimal routing cost R is constrained by

$$R \geq 100 - 20\, y_{23} - 10\, y_{45} \ . \tag{20}$$

That is, at best, the current routing cost which is 100 units will be reduced by the full savings of introducing arc (2, 3) (i.e., setting $y_{23} = 1$) and the full savings of introducing arc (4, 5) (i.e., setting $y_{45} = 1$).

Constraints like (20), which are known as Benders cuts, are by-products of the optimal routing calculation for any tentative network configuration. Benders algorithm computes the new configuration at each step by minimizing the fixed charge design cost F plus the routing cost bound R subject to the Benders cuts (20) generated by every previous tentative configuration. This minimization, called the Benders master problem, is a mixed integer program in the integer variables y_{ij} and single continuous variable R. At each step, one new constraint, a Benders cut, is added to the master problem. Note that since R becomes a lower bound on the routing cost, the minimum cost v of any design is bounded by the optimal value $F^* + R^*$ of the master problem, i.e., $v \geq F^* + R^*$. Also, every solution $y = \hat{y}$ to the master problem determines a network and the combined fixed and optimal routing cost on this network is an upper bound on v. These two bounds permit early termination of the algorithm with an assessment of degree of suboptimality.

When applied to more complicated design problems such as facility location problems, and even to general mixed integer programs, Benders decomposition operates in much the same way and has similar interpretations.

In their highly regarded paper [61] (see also [62]) concerning a facility location model with shipments from plants to customers through intermediate distribution centers, Geoffrion and Graves report on the most successful documented implementation of Benders decomposition to date. They solve problems with from 249 to 513 binary variables. From 0 to 30 of these correspond to distribution centers to be opened, or not; the remainder are 0-1 variables indicating whether or not a distribution center serves a customer. The computations required no more than seven Benders iterations to reach within 0.20% of the minimum cost design value, and required from 16.7 to 191 seconds of execution time on an IBM 360/91.

Their paper is rich in its description of implementation both in terms of management/model interaction and computer programming of Benders algorithm. The authors note, for example, that solving the Benders master problem to completion at each iteration may be an unnecessary computational burden. Rather, they search for a solution at each step that merely increases the lower bound by at least a given constant $\varepsilon > 0$. They also note that modeling constraints like $\sum_j x_{ij} \leq K_i y_i$, where $K_i = \sum_j d_j$, in the uncapacitated facility location model (see (14)) as $x_{ij} \leq K_i y_i$ for all i and j leads to much better algorithmic performance despite the fact that the latter representation requires many more constraints. We comment further on this observation in the next subsection.

Even for a given model representation, it is possible to accelerate Benders decomposition by generating "strong cuts" at each iteration. Referring to Figure 4 will help illustrate this point. Note that the shortest path distance from node 3 to node 6 using <u>all</u> arcs that are candidates for inclusion in the optimal network design is 60 units. Since the distance to node 2 in the current design, as specified by the dark arcs, is 20 units and the current shortest path cost is 100 units, introducing arc (2, 3) whose routing cost is 10 units can save no more than 100 - (60 + 20 + 10) = 10 units, and not the 20 units computed earlier. Consequently, a valid Benders cut is

$$R \geq 100 - 10\, y_{23} - 10\, y_{45} \,. \tag{21}$$

Note that this cut is stronger in the sense that it provides a tighter lower bound on R, than (20); the right-hand side of (21) is as large as that of (20) for all 0-1 values of the decision variables y_{ij}, and exceeds the right-hand side of (20) whenever $y_{23} = 1$.

The opportunity to generate strong cuts, like (21), is made possible because of degeneracy in the shortest path linear program, or equivalently because of multiple optimal solutions to its dual. In this example, $\pi_1 = 0$, $\pi_2 = 20$, $\pi_3 = 40$, $\pi_4 = 70$, $\pi_5 = 90$, and $\pi_6 = 100$ is an alternate optimal dual solution to that shown in Figure 4. Computing a Benders cut as before, but using these dual values leads to the stronger cut (21). Because network problems are renowned for their degeneracy, considerations of this nature are attractive for a number of transportation applications.

Magnanti and Wong [96] describe this strong cut methodology in greater detail. They show how to generate pareto-optimal cuts, i.e., cuts with the property that no other is stronger, for arbitrary mixed integer programs by

solving a linear program to choose from among optimal dual solutions. Specializations of this approach lead to a pareto-optimal cut methodology for P-median problems that avoids explicit linear programming computations. Computational experience on a variety of P-median problems (up to 33 nodes) shows that Benders algorithm equipped with strong cuts finds solutions known to be within 10% of optimality in ten or fewer iterations. The standard implementation usually provides no better solutions within 25 iterations and solutions 10% farther from optimality within ten iterations. The authors obtain similar experience with strong cuts for uncapacitated fixed charge design problems, though in this case the error bounds are generally not as tight.

Heuristics

Several researchers (Billheimer and Gray [16], Scott [111], Stairs [117], Steenbrink [118], and Dionne and Florian [38] among others) have proposed heuristic procedures for solving network design problems. These are generally of three types—add, delete, or interchange. The add heuristics start with some feasible design and add arcs, one at a time, choosing at each stage the arc that gives the greatest decrease in cost, or some surrogate measure of cost. The delete heuristics are similar, but start with an initial design containing all candidate arcs, and delete arcs one at a time. Starting with some initial design, the interchange heuristics add and/or delete an arc at each step until no further improvement in cost is possible.

Recently, Dionne and Florian [38] have reported impressive computational experience with a new type of heuristic for budget design problems with, for convenience, one unit of demand between every two nodes in the network. They use their branch and bound algorithm described earlier in this section with the following modification. In place of the term $I_{ij}(y^P)$ in the lower bound expression (16), they use a term $I(y^P)$ which is the increment to shortest route costs between every pair of nodes, not just i and j, when arc (i,j) is deleted from the network. This algorithm is a heuristic because, as examples show, the new lower bound need not be valid.

The authors have tested this algorithm on problems ranging from 7 nodes and 16 candidate arcs to 29 nodes and 54 candidate arcs, and they have compared its performance with an add heuristic, with Scott's [111] combined delete and exchange heuristic, and with a variant of this algorithm. With but one exception, in which the relative error between the heuristic and optimal solutions was 0.03%, the new heuristic found the optimal solution to every problem. The

relative errors ranged from about 1% to 7% for the add heuristic and were less than 1% in all cases for the other heuristics. The new heuristic required from 0.05 to 8.47 seconds of computation time on a CDC Cyber 74 computer.

Cornuejols et al. [28], who cite a number of references on heuristics for facility location models, have initiated a new line of investigation. They consider uncapacitated fixed charge facility location models with the P-median constraint $\Sigma\, y_i \leq P$. Let us consider the special case where $f_i = 0$, $d_j = 1$ and $c_{ij} \geq 0$ for all i and j. These assumptions are not essential, but are merely convenient for our exposition. To conform with this paper, we assume that the problem has been formulated in maximization form.

The authors derive the following results. If v is the optimal value to the problem and v^a is the value of the solution determined by the add heuristic then

$$\frac{v - v^a}{v} \leq \left(\frac{P - 1}{P}\right)^P < \frac{1}{e} \tag{22}$$

where e is the base of the natural logarithm. The clever proof of this result uses the fact that the value v^D of the Lagrangean dual problem formed by dualizing with respect to the demand constraints $\sum_i x_{ij} = 1$ is equivalent to the optimal value of the linear programming version of the problem (i.e., $0 \leq y_i \leq 1$) when the constraints $x_{ij} \leq y_i$ for all i and j replace (14). The analysis of the Lagrangean dual shows that (22) is valid when v^D replaces v. Since $v^a \leq v \leq v^D$, we also have

$$\frac{v^D - v}{v^D} \leq \left(\frac{P - 1}{P}\right)^P < \frac{1}{e} \tag{23}$$

which indicates by how much the value of the linear programming relaxation of the problem can deviate from the value of the problem itself.

Not only do the authors show, by examples, that these bounds are best possible, they also construct examples to show that other heuristics (delete, exchange, dynamic programming) cannot provide relative error bounds as tight as (22). Moreover, they show that the relative error of the optimal value of the "weak" linear programming relaxation based upon the constraints (14), rather than the "strong" formulation based upon the constraints $x_{ij} \leq y_i$ for all i and j, need not be as tight as (23) (the relative error can be made as close to 1 as desired by judicious choice of data). This fact helps to explain modeling

experience mentioned previously, namely that the strong linear programming formulation is preferred to the weak formulation.

In computational experiments with problems containing as many as 164 potential locations for facilities, the authors used the add heuristic followed by a subgradient algorithm applied to the Lagrangean dual to obtain an improved feasible solution and upper bound. This algorithm determined the optimal solution to almost every problem that was tested and required no more than 30 seconds of computation time on an IBM 370/168 computer.

7. Open Research Problems

In this section, we enumerate several areas where future work may prove fruitful:

* Aggregation of Network Data. In most transportation studies, underlying data is aggregated in order to obtain data sets that are manageable from both the viewpoints of data collection and limitations on algorithmic capabilities. What are the "best" procedures for aggregating data, how are aggregate solutions to be disaggregated for the actual problem setting, and what degree of suboptimality does the aggregation/disaggregation process entail? Chan et al. [23], Geoffrion [58], [59], Kuhn [91], and Kuhn and Cullum [92] have initiated research to answer these questions.

* Shortest Paths from an Origin to a Few Destinations. There are very efficient algorithms for finding shortest paths from an origin to one other node and to all other nodes. In practice, however, it might be desired that several nodes serve as destinations. Can this problem be approached directly? In particular, when distances are Euclidean can the geometry be exploited to obtain more efficient algorithms?

* Multicommodity Flow. Can list processing structures, like those recently developed for transshipment problems (see section 3), be extended efficiently to solve multicommodity flow problems with bundle constraints? Are there effective heuristic techniques for large multicommodity flow problems?

* Normative Models of Urban Traffic Flow. Suppose that a system optimized traffic equilibrium (e.g., fuel minimization) is desired, but that users act in accordance with Wardrop's first principle of individual cost, or time, minimization. What available policy alternatives, e.g., one-way street assignments, prohibited turns, or road tolls, guarantee that a user equilibrium would coincide with the desired system equilibrium? Rosenthal [108] has shown that road tolls, alone, will not suffice. To perform analysis of this nature might require solution methodologies which incorporate predictive models as part of the underlying constraint structure of normative optimization. Theoretical tools for performing sensitivity analysis on equilibrium solutions (see Hall [71]) would undoubtedly aid this effort, and would be of independent interest to transportation planners.

* Predicting Origin and Destination Choice of Urban Traffic. When the traffic equilibrium model includes destination choice possibilities (see section 4) there is no known equivalent convex optimization problem for predicting traffic flow. Origin choice models that would be capable of predicting homeowner location decisions for long range urban planning encounter the same difficulty. The ability to compute solutions to either variant of this model, or to a combined origin and destination choice model, would greatly extend urban traffic planning capabilities.

* The Stochastic Vehicle Routing Problem. There has been some recent work on this problem as described in this paper, but much more research needs to be done and real applications need to be investigated.

* The Vehicle Routing Problem with Window Constraints. Timing restrictions become a component of the vehicle routing model in the event that some customers impose delivery deadlines and earliest delivery time constraints, thereby imposing a "window" or "interval" of time during which a delivery or pickup must be made. Biles and Bradford [15] and Russell [109] have recently looked at these sequencing restrictions, but more efficient algorithms should be within reach.

* Tight Lower Bounds for Vehicle Routing Problems. Can a relaxation approach similar in nature to the work of Held and Karp [76], [77] be of value in deriving tight lower bounds on the minimum distance solution to the vehicle routing problem. This would allow one to evaluate the effectiveness of various heuristic approaches.

* Airline Crew Scheduling. An airline has a set of m flights each of which requires one of a set of N crews. Crews must be allocated to flights in order to minimize operating costs (see Arabeyre et al. [5] for details). Powerful heuristics need to be developed and analyzed for this important class of problems.

* Worst-Case Analysis of Heuristic Algorithms. For a given network optimization problem where the only known efficient solution procedures are heuristics, how badly might the heuristic solution deviate from the optimal solution? Rosenkrantz et al. [106] and Cornuejols et al. [28] provide examples of this kind of analysis.

* Probabilistic Analysis of Network Algorithms. See Karp [82] for a number of open problems of a more theoretical nature relating to probabilistic, as opposed to worst-case, analysis of heuristics.

* The Loading Problem. Let x_{ij} and y_{ij} denote the flow of goods and vehicles on the arcs (i,j) of the same network. Given a linear or nonlinear objective function of $x = (x_{ij})$ and $y = (y_{ij})$ to be minimized, add to the separate transshipment constraints for goods and vehicles loading constraints of the form $x_{ij} \leq K\, y_{ij}$ for each arc (i,j). The interpretation is that goods are to be loaded on vehicles each with capacity K. Applications, among others, are the loading of passengers on planes, the loading of cargo on trains, and the assignment of railcars to rail engines. Straightforward modeling extensions encompassing multicommodity goods and multiple vehicle types might be expected in practice. What is an efficient solution technique for this class of problems?

* Freight Flow Management. Modeling of freight involves a number of issues. In rail applications, decisions must be made concerning the assignment of freight to cars, the composition (in terms of cars) of trains leaving a railyard, train routing, the scheduling of engines to train routes, the capacity and location of railyards, and many other aspects of rail operations. Modeling is complicated by "blocking" or "grouping" contingencies in which cars destined for several final locations are grouped together and treated as a unit along their route to some intermediate railyard. In most instances, the number of possibilities for blocking and "reblocking" is enormous.

Although analytic approaches have been successful for dealing with certain aspects of freight flow management (Assad [8] delineates efforts in rail), there remains great potential for modeling and algorithmic development.

1. Aashtiani, H., "Multi-Modal Traffic Assignment," Ph.D. Thesis, Operations Research Center, M.I.T. (in preparation).
2. Aashtiani, H. and Magnanti, T., "Implementing Primal-Dual Network Flow Algorithms," Working Paper OR 055-76, Operations Research Center, M.I.T. (June 1976).
3. Abdulaal, M. and Leblanc, L., "Multimodal Network Equilibrium," Tech. Report IEOR 77013, Dept. of Ind. Eng. and Opers. Res., Southern Methodist Univ. (June 1977).
4. Amit, I. and Goldfarb, D., "The Timetable Problem for Railways," 379-387, in *Developments in Operations Research*, Vol. 2 (B. Avi-itzhak, ed.), Gordon and Breach, New York (1971).
5. Arabeyre, J., Fearnley, J., Steiger, F., and Teather, W., "The Airline Crew Scheduling Problem: A Survey," *Trans. Sci.*, 3, 140-163 (1969).
6. Assad, A. A., "Multi-commodity Network Flows--A Survey," forthcoming in *Networks*.
7. Assad, A. A., "Multi-commodity Network Flows--Computational Experience," Working Paper OR 058-76, Operations Research Center, M.I.T. (Oct. 1976).
8. Assad, A. A., "Supply Modelling of Rail Networks: Toward a Routing/Makeup Model," Working Paper OR 069-77, Operations Research Center, M.I.T. (Dec. 1977).
9. Baratz, A., "Construction and Analysis of Network Flow Problem Which Forces Karzanov Algorithm to $O(N^3)$ Running Time," Tech. Memo LCS/TM-83, Laboratory for Computer Science, M.I.T. (April 1977).
10. Barr, R., Glover, F., and Klingman, D., "Enhancements of Spanning Tree Labeling Procedures for Network Optimization," Res. Report CCS 262, Center for Cybernetic Studies, Univ. of Texas (Dec. 1976).
11. Beckman, M. J., McGuire, C. B. and Winsten, C. B., *Studies in the Economics of Transportation*, Yale Univ. Press, New Haven, CT (1956).
12. Bellman, R., "On a Routing Problem," *Quart. of Applied Mathematics*, 16, 87-90, (1958).
13. Beltrami, E. and Bodin, L., "Networks and Vehicle Routing for Municipal Waste Collection," *Networks*, 4, (1), 65-94 (1974).
14. Bennett, B. and Gazis, D., "School Bus Routing by Computer," *Transportation Research*, 6, (4), 317-325 (1972).
15. Biles, W. and Bradford, J., "A Heuristic Approach to Vehicle Scheduling with Due-Date Constraints," presented at the Spring ORSA/TIMS Meeting, Chicago, Illinois (1975).

16. Billheimer, J. and Gray, P., "Network Design with Variable Cost Elements," *Trans. Sci.*, 7, 49-74 (1973).

17. Bodin, L., "A Taxonomic Structure for Vehicle Routing and Scheduling Problems," *Comput. and Urban Soc.*, 1, 11-29 (1975).

18. Boyce, D. E., Farhi, A., and Weischedel, R., "Optimal Network Design Problem: A Branch and Bound Algorithm," *Environment and Planning*, 15, 519-533, (1973).

19. Bradley, G., "Survey of Deterministic Networks," *AIIE Transactions*, 7, (3), 224-234 (1975).

20. Bradley, G. H., Brown, G. G., and Graves, G. W., "Design and Implementation of Large Scale Primal Transshipment Algorithms," *Management Science*, 24(1), 1-34, 1977.

21. Branston, D., "Link Capacity Functions: A Review," *Trans. Res.*, 10, 223-236 (1976).

22. Bruynooghe, M., "Un Modèle Intégré de Distribution et d'Affection de Traffic sur un Réseau," Tech. Report, Inst. de Recherche sur les Transports, Dept. de Recherche Opérationelle et Informatique, Arcueuil (1969).

23. Chan, Y., Follansbee, K. G., Manheim, M. L., Mumford, J. R., "Aggregation in Transportation Networks: An Application of Hierarchical Structure," Report R68-47, Dept. of Civil Eng., M.I.T. (1968).

24. Chen, D. and Zionts, S., "Comparison of Some Algorithms for Solving the Group Theoretic Integer Programming Problem," *Operations Research*, 24, 1120-1128 (1976).

25. Christofides, N., *Graph Theory: An Algorithmic Approach*, Academic Press, New York (1975).

26. Christofides, N. and Brooker, P.,"Optimal Expansion of an Existing Network," *Math. Prog.*, 6, 197-211 (1974).

27. Clarke, G. and Wright, J., "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," *Operations Research*, 12 (4), 568-581 (1964).

28. Connuejols, G., Fisher, M., and Nemhauser, G., "Location of Bank Accounts to Optimize Float: An Analytical Study of Exact and Approximate Algorithms," *Management Science*, 23 (8), 789-810 (1977).

29. Dantzig, G. B., Harvey, R. P., Lansdowne, Z. F., Maier, S. F., McKnight, R. W., and Robinson, D. W., "Two Geographic Decomposition Approaches in Transportation Network Analysis," Tech. Report, Control Analysis Corp., Palo Alto, California (June 1977).

30. Dantzig, G. B., Maier, S. F. and Lansdowne, Z. F., "Application of Decomposition to Transportation Network Analysis," Tech. Report DOT-TSC-OST-76-26, Control Analysis Corporation (Oct. 1976).

31. Davis, P. S. and Ray, T. L., "A Branch-Bound Algorithm for the Capacitated Facilities Location Problem," *Naval Res. Log. Quart.*, 16, 331-334, (1969).

32. de Ghellinck, G., private communication, May 1977.

33. Denardo, E. and Fox, B., "Shortest Route Methods: 1. Reaching, Pruning and Buckets," forthcoming in *Operations Research*.

34. Denardo, E. and Fox, B., "Shortest Route Methods: 2. Group Knapsacks, Expanded Networks, and Branch and Bound," forthcoming in *Operations Research*.

35. Dial,R. B., "A Combined Trip Distribution and Modal Split Model," paper presented at the 1974 Annual Meeting of the Highway Research Board (1973).

36. Dijkstra, E., "A Note On Two Problems in Connection with Graphs," *Num. Mathematik*, 1, 269-271 (1959).

37. Dinic, E. A., "Algorithm for Solution of a Problem of Maximum Flow in a Network with Power Estimation," *Soviet Math. Dokl.*, 11, 1277-1280 (1970).

38. Dionne, R., and Florian, M., "Exact and Approximate Algorithms for Optimal Network Design," Publ. #41, Centre de Recherche sur les Transports, Univ. of Montreal (Feb. 1977).

39. Domenich, T., and McFadden, D., *Urban Travel Demand*, North Holland Publ. Co., Amsterdam (1975).

40. Dreyfus, S., "An Appraisal of Some Shortest Path Algorithms," *Operations Research*, 17, 395-412 (1969).

41. Edmonds, J., and Karp, R. M., "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems," *JACM*, 19,248-264 (1972).

42. Efroymson, M. A and Ray, T. L., "A Branch and Bound Algorithm for Plant Location," *Operations Research*, 14, 361-368 (1966).

43. Evans, S., "Some Models for Combining the Trip Distribution and Traffic Assignment Stages in the Transport Planning Process," in *Traffic Equilibrium Methods, Proceedings 1974* (M. Florian, ed.) Vol. 118, Lecture Notes in Economics and Mathematical Systems, Springer Verlag (1976).

44. Even, S., "The Max Flow Algorithm of Dinic and Karzanov," Tech. Memo LCS/TM-80, Laboratory for Computer Science, M.I.T. (Dec. 1976).

45. Florian, M., "A Traffic Equilibrium Model of Travel by Car and Public Transit Modes," forthcoming in *Trans. Sci.*

46. Florian, M. and Nguyen, S., "An Application and Validation of Equilibrium Trip Assignment Methods," *Trans. Sci.*, 10, 374-390 (1976).

47. Florian, M. and Nguyen, S., "Multicommodity Flow Problems with Convex Costs: Models, Algorithms, and Applications," Centre de Recherche sur les Transports, University of Montreal (March 1977).

48. Florian, M. and Nguyen, S., "A Combined Trip Distribution, Modal Split and Trip Assignment Model," Publication 34, Centre de Recherche sur les Transports, Univ. of Montreal (March 1977).

49. Florian, M., Nguyen, S., and Ferland, J., "On the Combined Distribution Assignment of Traffic," *Trans. Sci.*, 9, 43-53 (1975).

50. Floyd, R., "Algorithm 97--Shortest Path," *CACM*, 5, 345 (1962).

51. Frank, H. and Frisch, I., *Communication, Transmission, and Transportation Networks*, Addison-Wesley, Reading, MA (1971).

52. Frieze, A., "Shortest Path Algorithms for Knapsack Type Problems," *Math. Prog.*, 11, 150-157 (1976).

53. Gartner, N., Golden, B., and Wong, R., "Modelling and Optimization for Transportation Systems Planning and Operations," *Proc. of the Intern. Symp. on Large Engineering Systems*, Winnipeg, Canada, August (1976).

54. Gartner, N., Little, J. D. C., and Gabbay, H., "Optimization of Traffic Signal Settings by Mixed- Integer Linear Programming, Parts I and II," *Trans. Sci.*, 9,321-363 (1975).

55. Garvin, W., Crandall, H., John, J., and Spellman, R., "Applications of Linear Programming in the Oil Industry," *Management Science*, 3 (4), 407-430 (1957).

56. Gazis, D., "Transportation Networks," *Networks*, 5 (1) 75-79 (1975).

57. Geoffrion, A., "Lagrangean Relaxation for Integer Programming," *Math. Prog. Study #2*, 82-114 (1974).

58. Geoffrion, A. M., "A Priori Error Bounds for Procurement Commodity Aggregation in Logistics Planning Models," *Naval Res. Log. Quart.*, forthcoming 1977.

59. Geoffrion, A. M., "Customer Aggregation in Distribution Modeling," Working Paper No. 259, Western Man. Sci. Inst., UCLA (Oct. 1976).

60. Geoffrion, A. M., "Better Distribution Planning with Computer Models," *Harvard Business Review* 54(4), 92-99 (1976). Reprinted in this volume.

61. Geoffrion, A. M. and Graves, G., "Multicommodity Distribution Systems Design by Benders Decomposition," *Man. Sci.*, 20, 822-844 (1974).

62. Geoffrion, A. M., Graves, G., and Lee, S., "Strategic Distribution System Planning: A Status Report," in *Studies in Operations Management* (A. C. Hax, ed.) North Holland-American Elsevier (1978).

63. Geoffrion, A. M. and McBride, R., "Lagrangean Relaxation Applied to Facility Location Problems," Working Paper 263, Western Man. Sci. Inst., UCLA (Jan. 1977).

64. Gillett, B. and Miller, L., "A Heuristic Algorithm for the Vehicle Dispatch Problem," *Operations Research*, 22, 340-349 (1974).

65. Gilsinn, J. and Witzgall, C., "A Performance Comparison of Labeling Algorithms for Calculating Shortest Path Trees," NBS Technical Note 777, National Bureau of Standards, Washington, DC (1973).

66. Golden, B., "Shortest-Path Algorithms: A Comparison," *Operations Research*, 24, 1164-1168 (1976).

67. Golden, B. and Ball, M., "Shortest Paths with Euclidean Distances: An Explanatory Model," submitted for publication.

68. Golden, B. and Magnanti, T., "Deterministic Network Optimization--A Bibliography," *Networks*, 7 (2), 149-183 (1977).

69. Golden, B., Magnanti, T., and Nguyen, H., "Implementing Vehicle Routing Algorithms," *Networks*, 7 (2), 113-148 (1977).

70. Golden, B. and Stewart, W., "Vehicle Routing with Probabilistic Demands," *Proc. of the Tenth Annual Symposium on the Interface of Computer Science and Statistics*, Gaithersburg, Maryland, April 14-15 (1977).

71. Hall, M. A., "Properties of the Equilibrium State in Transportation Networks," Unpublished Report, Bell Laboratories, Piscataway, NJ (1976).

72. Hall, M., "New Methods for Computing Transportation Network Equilibria," ORSA/TIMS Bulletin, 2, p. 212 (Nov. 1976).

73. Hart, P., Nilsson, N., and Raphael, B., "A Formal Basis for the Heuristic Determination of Minumum Cost Paths," *IEEE Trans. on Systems Science and Cybernetics, Vol. SSC-4*, 100-107 (1968).

74. Harvey, R. P. and Robinson, D. W., "Computer Code for Transportation Network Design and Analysis," Tech. Report DOT-TSC-OST-77-36, Control Analysis Corporation (May 1977).

75. Hax, A., "Aggregate Production Planning," in *Handbook of Operations Research* (J. Moder and S. E. Elmaghraby, eds.) Van Nostrand Reinhold, New York, forthcoming.

76. Held, M. and Karp, R., "The Traveling Salesman Problem and Minimum Spanning Trees," *Operations Research*, 18 (6), 1138-1162 (1970).

77. Held, M. and Karp, R., "The Traveling Salesman Problem and Minimum Spanning Trees: Part II," *Math. Prog.*, 1, 6-25 (1971).

78. Hern, D. W., "Network Aggregation in Transportation Planning, Part I," Tech. Report, Mathtech, Inc., Princeton, NJ (June 1977).

79. Hoang, H. H., "A Computational Approach to the Selection of An Optimal Network," *Man. Sci.*, 19, 488-498 (1973).

80. Houpt, P. K., and Athans, M., "Dynamic Stochastic Control of Freeway Corridor Systems: Volume I--Summary," Report ESL-R-608, Electronics Systems Laboratory, M.I.T. (Aug. 1975).

81. Johnson, D. S., Lenstra, J. K. and Rinnooy Kan, A. H. G., "The Complexity of the Network Design Problem," Mathematisch Centrum Amsterdam (1976).

82. Karp, R., "The Probabilistic Analysis of Some Combinatorial Search Algorithms," in *Algorithms and Complexity: Recent Results and Directions*, 1-19 (J. F. Traub, ed.), Academic Press (1976).

83. Karzanov, A. V., "Determining the Maximal Flow in a Network by the Method of Preflows," *Soviet Math. Dokl.*, 15, 434-437 (1974).

84. Kennington, J., "Multi-commodity Flows: A Survey of Linear Models and Solution Techniques," forthcoming in *Operations Research*.

85. Kennington, J. L., "Solving Multicommodity Transportation Problems Using A Primal Partitioning Simplex Technique," Report CP 75013, Dept. of Ind. Eng. and Oper. Res., Southern Methodist Univ. (revised May 1976).

86. Kennington, J. L. and Shalaby, M., "An Effective Subgradient Procedure for Minimal Cost Multicommodity Flow Problems," *Management Science*, 23, 994-1004 (1977).

87. Kershenbaum, A., Hsieh, W., and Golden, B., "Constrained Routing in Large Sparse Networks," *Conference Records of 1976 IEEE Intern. Conference on Communications*, 38.14-38.18, Philadelphia, June 14-16 (1976).

88. Kirby, R. and Potts, R., "The Minimum Route Problem for Networks with Turn Penalties and Prohibitions," *Trans. Res.*, 3, 397-408 (1969).

89. Klingman, D., Dial, R., Glover, F., and Karney, D., "Solving Large Scale Shortest Path Problems," TIMS/ORSA Bulletin, 147 (May 1977).

90. Kuehn, A. A. and Hamburger, M. J., "A Heuristic Program for Locating Warehouses," *Man. Sci.*, 9, 643-666 (1963).

91. Kuhn, H., "Network Aggregation in Transportation Planning, Part II," Tech. Report, Mathtech, Inc., Princeton, NJ (June 1977).

92. Kuhn, H. W. and Cullum, D. E., "Aggregation in Network Models for Transportation Planning," Tech. Report, Mathtech, Princeton, NJ (1976).

93. Leblanc, L., "An Algorithm for the Discrete Network Design Problem," *Trans. Sci.*, 9, 183-187 (1975).

94. Magnanti, T., "Optimization for Sparse Systems," in *Sparse Matrix Computations* (J. R. Bunch and D. J. Rose, eds.), Academic Press, New York, 147-176 (1976).

95. Magnanti, T., Shapiro, J., and Wagner, M., "Generalized Linear Programming Solves the Dual," *Management Science*, 22, 1195-1203 (1976).

96. Magnanti, T. L. and Wong, R. T., "Accelerating Benders Decomposition for Network Design," Discussion Paper, Center for Operations Research and Econometrics, Catholique Univ. de Louvain, Belgium (1978).

97. Mulvey, J., "Testing of a Large-Scale Network Optimization Program, Harvard Working Paper HBS 75-38 (1975), submitted to *Math. Prog.*

98. Nemhauser, G., "A Generalized Permanent Label Setting Algorithm for the Shortest Path Between Specified Nodes," *J. of Math. Anal. and Appl.*, 38, 328-334 (1972).

99. Nguyen, S., "A Mathematical Programming Approach to Equilibrium Methods of Traffic Assignment with Fixed Demands," Publ. #138, Dept. Informatique, Univ. of Montreal (1973).

100. Nguyen, S., "Procedures for Equilibrium Traffic Assignment with Elastic Demand," Publication 39, Centre de Recherche sur les Transports, Univ. of Montreal (revised March 1977).

101. Nguyen, S., "On the Estimation of An OD Trip Matrix by the Equilibrium Methods Using Pseudo Delay Functions," Unpublished Report, Centre de Recherche sur les Transports, Univ. of Montreal (1977).

102. Nguyen, S., "Estimating an OD Matrix from Network Data: A Network Equilibrium Approach," Publication #60, Centre de Recherche sur les Transports, Univ. of Montreal (Feb. 1977).

103. Orloff, C., "Routing a Fleet of M Vehicles to/from a Central Facility," *Networks*, 4 (2), 147-162 (1974)

104. Pape, U., "Implementation and Efficiency of Moore-Algorithms for the Shortest Route Problem," *Math. Prog.*, 7, 212-222 (1974).

105. Potts, R. and Oliver, R., *Flows in Transportation Networks*, Academic Press, New York (1972).

106. Rosenkrantz, D., Stearns, R., and Lewis, P., "Approximate Algorithms for the Traveling Salesperson Problem," *Proc. of the 15th Annual IEEE Symposium On Switching and Automata Theory*, 33-42 (1974).

107. Rosenthal, R. W., "The Network Equilibrium Problem in Integers," *Networks*, 3, 53-59 (1973).

108. Rosenthal, R. W., "Congestion Tolls: Equilibrium and Optimality," Economic Discussion Paper #94, Bell Laboratories (April 1977).

109. Russell, R., "An Effective Heuristic for the M-tour Traveling Salesman Problem with Some Side Conditions," *Operations Research*, 25 (3), 517-524 (1977).

110. Salzborn, F. J. N., "Timetables for a Suburban Rail Transit System," *Trans. Sci.*, 3, 297-316 (1969).

111. Scott, A. J., "The Optimal Network Problem: Some Computational Procedures," *Trans. Sci.*, 3, 201-210 (1969).

112. Scott, A., *Combinatorial Programming, Spatial Analysis and Planning*, Methuen, London (1971).

113. Shapiro, J., "Shortest Route Methods for Finite State Space Deterministic Dynamic Programming Problems," *SIAM J. on Applied Math.*, 16, 1232-1250 (1968).

114. Shier, D., "Computational Experience with an Algorithm for Finding the K Shortest Paths in a Network," *J. of Research of NBS, 78B*, 139-165 (1974).

115. Shier, D., "Iterative Methods for Determining the K Shortest Paths in a Network," *Networks*, 6, 205-230 (1976)

116. Simpson, R., "Scheduling and Routing Models for Airline Systems," M.I.T. Flight Transportation Laboratory Report, December (1969).

117. Stairs, S., "Selecting a Traffic Network," *Journal of Transport Economics and Policy*, 2, 218-231 (1968).

118. Steenbrink, P., *Optimization of Transport Networks*, Wiley, London (1974).

119. Stewart, W., "The Delivery Truck Routing Problem with Stochastic Demands," Management Science/Statistics Working Paper, MS/S 76-005, University of Maryland at College Park (1976).

120. Swoveland, C., "Decomposition Algorithms for the Multicommodity Distribution Problem, Working Paper #184, Western Man. Sci. Inst., UCLA (Oct. 1971).

121. Turner, W., Ghare, P., and Fourds, L., "Transportation Routing Problem--A Survey," *AIIE Transactions*, 6 (4), 288-301 (1974).

122. Wardrop, J. G., "Some Theoretical Aspects of Road Traffic Research," *Proc. Inst. Civil Engineers, Part II*, 1, 325-378 (1952).

123. Weintraub, A., "Optimal Flows in Networks and Games: The Multicommodity Flow Problem in Integers," Publication 76/21/C, Dept. of Ind. Eng., Univ. of Chile (Oct. 1976).

124. White, W. W. and Bomberault, A. M., "A Network Algorithm for Empty Freight Car Allocation," *IBM Systems Journal*, 9 (2), 147-169 (1969).

125. Wong, R. T., "A Survey of Network Design Problems," Working Paper OR 053-76, Operations Research Center, M.I.T. (May 1976).

126. Zadeh, N., "More Pathological Examples for Network Flow Problems," *Math. Prog.*, 5, 217-224 (1973).

127. Zadeh, N., "A Bad Network Problem for the Simplex Method and Other Minimum Cost Flow Algorithms," *Math. Prog.*, 5, 255-266 (1973).